Colección Electrónica

PROGRAMACION DE COMPUTADORAS

Programas en BASIC

NO NECESITA COMPUTADORA

Ediciones Plesa

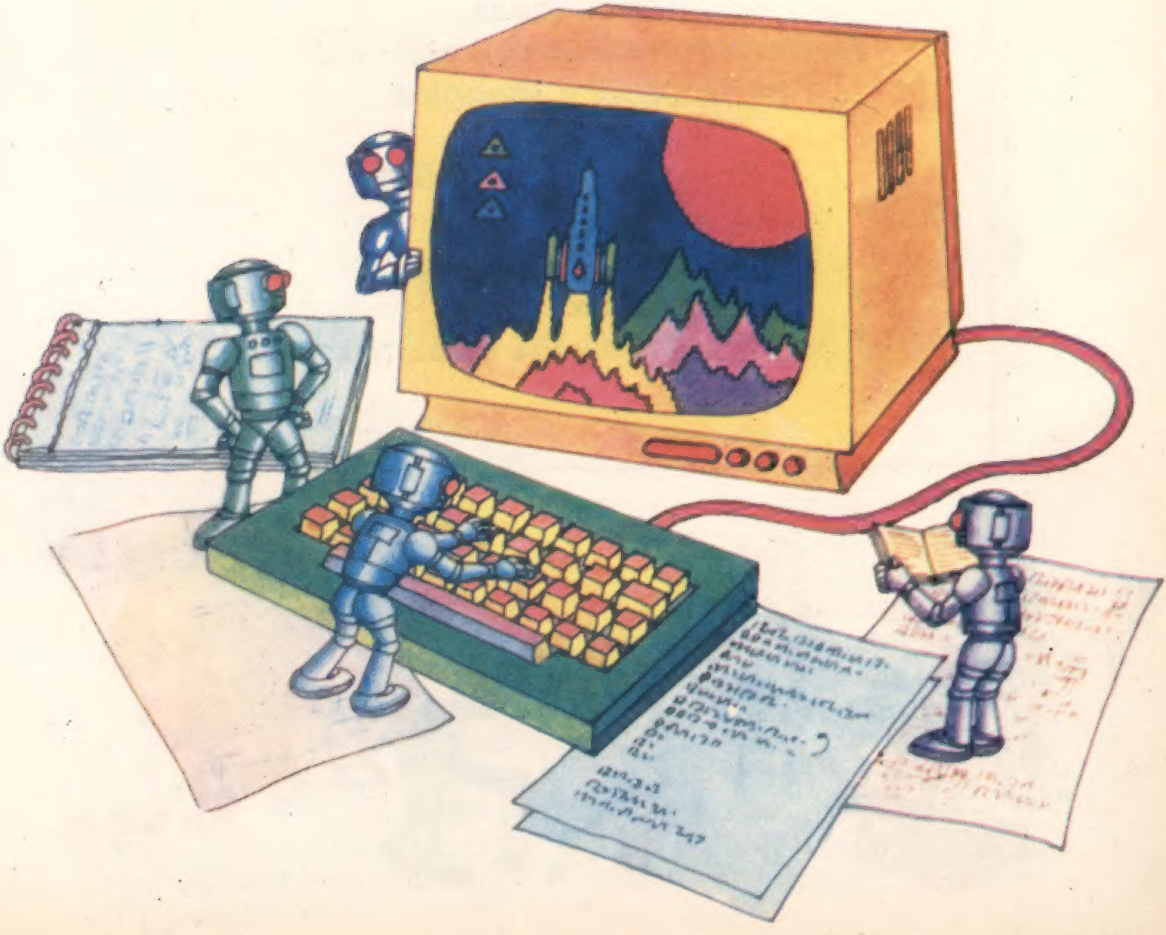
Colección Electrónica

PROGRAMACION DE COMPUTADORAS

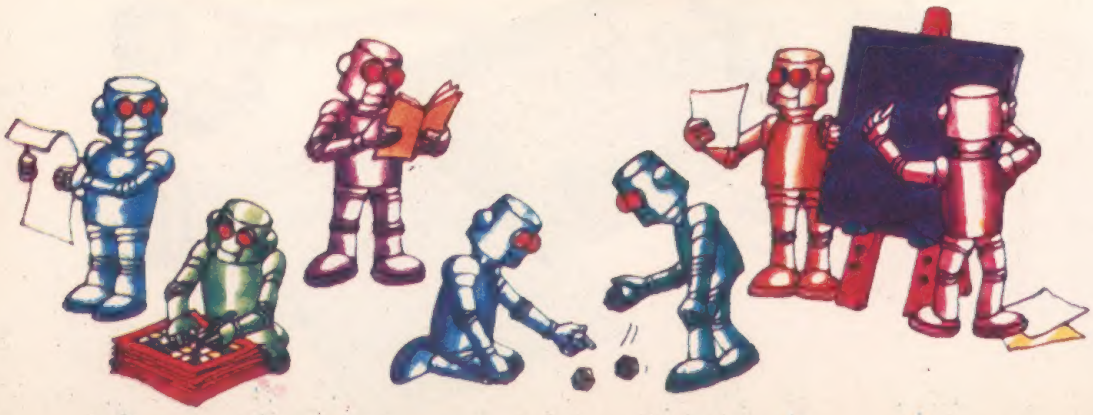
Brian Reffin Smith

Ilustrado por Graham Round y Martin Newton

Adaptado por Antonio Zorita García



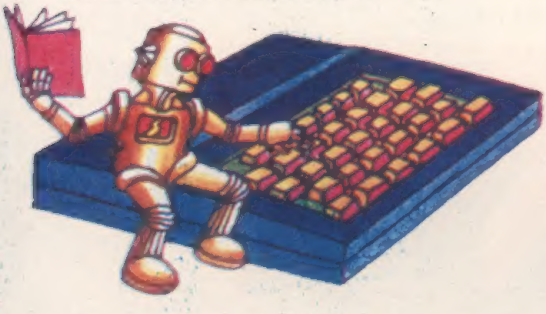
Contenido



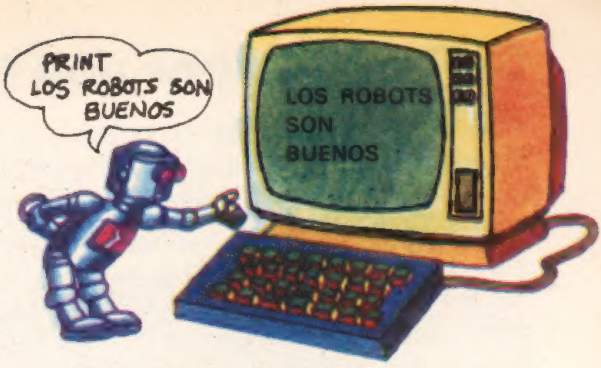
Sobre este libro

Esta es un guía de iniciación para escribir programas de computadora en el sistema llamado *BASIC*. Este es el lenguaje utilizado por la mayoría de las computadoras personales.

No es necesaria una computadora para usar este libro, aunque sería muy útil para comprender los programas si los pruebas al mismo tiempo. Diferentes computadoras usan ligeras variantes de *BASIC*; todos los términos de este libro pertenecen a la terminología standard, y cuando esto no es así, se indicará claramente.



Al principio hay algunas líneas para programar una computadora. Después, las principales palabras de *BASIC* aparecerán una por una con cortos programas para mostrar cómo funcionan.



Para obtener práctica en escribir programas hay preparados problemas y sugerencias que permiten alteraciones útiles en los programas propuestos en este libro. Las respuestas a los problemas o puzzles están en las páginas 44-45.

Al final del libro hay una lista de términos *BASIC* y otras palabras de computadoras con breves explicaciones. Hay ayudas para conseguir escribir tus propios programas y una lista de "bugs" con pistas que te ayuden a reconocerlos.

Si tienes una "micro" prueba los programas de esta guía, y luego para aprender más usos de tu computadora, busca los términos de *BASIC* en tu manual. Observarás que algunas de las normas que aquí aparecen no son necesarias en tu computadora. Pero la mejor forma de aprender es probar los programas que aparecen en los libros y revistas, y luego alterarlos un poco para saber qué sucede. A partir de aquí te será fácil escribir tus propios programas.

Cómo funciona una computadora

Puedes hacer muchas cosas con una computadora. Puedes dibujar, escribir una divertida poesía, dibujar gráficos, jugar y entretenerte con palabras y números, algunas veces de forma útil y otras sólo por diversión. Una computadora se describe algunas veces con un "procesador de información". Su misión consiste en tomar la información que le das, trabajar con ella según tus instrucciones y darte los resultados

Para lograr que una computadora haga lo que tú quieras, tienes que darle instrucciones muy detalladas. Esta lista de instrucciones se llama programa y la información que tú le das a la computadora se llama «data». El programa tiene que ser escrito en un código tal como *BASIC*, que la computadora puede comprender; es necesario seguir también las reglas del código.

Microcomputadores

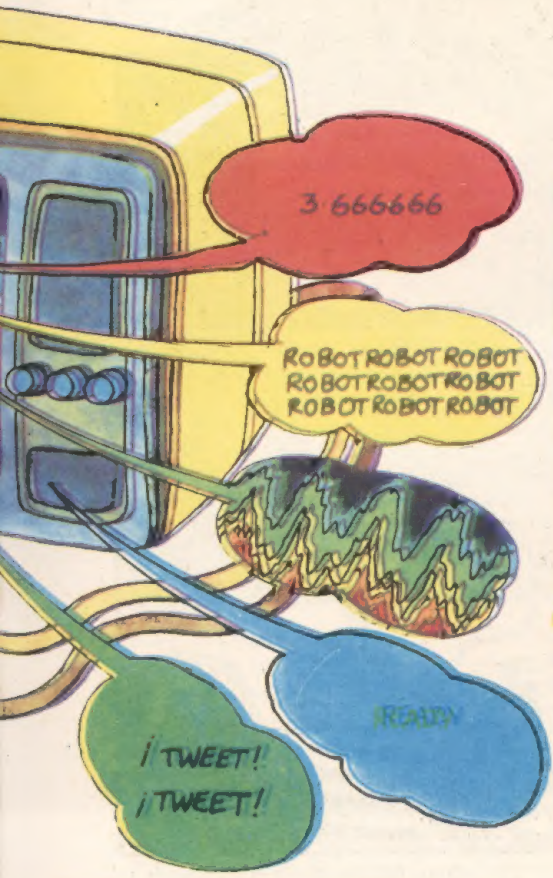
La mayor parte de los micros están formados por un teclado que puedes conectar en un aparato de TV. A la computadora le das instrucciones e información tecleando sobre el tablero, y todo lo que tecleas aparece en la pantalla de TV.

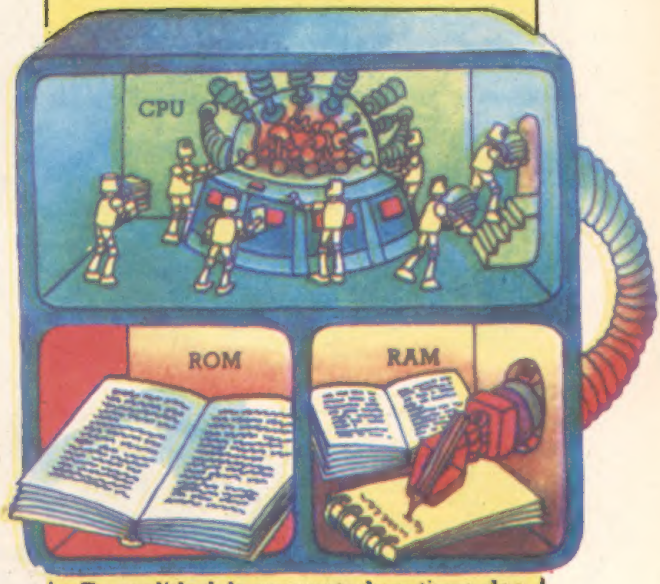
Algunos micros tienen una pantalla pequeña construida en el propio aparato, como las calculadoras. Otros tienen un pantalla especial llamada monitor, que equivale a una TV, pero no puede recibir señales de una emisora de TV.

El teclado de un micro es como el de una máquina de escribir con algunas teclas especiales. En algunos micros ciertas claves dan instrucciones especiales a la computadora en *BASIC*, de forma que no es preciso teclear las palabras letra por letra.

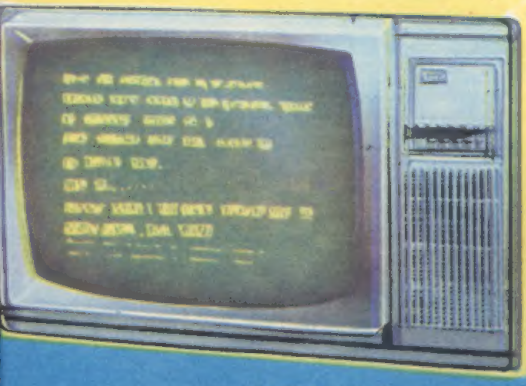
Interior de un micro

Un micro tiene dos partes fundamentales: la unidad central de proceso (CPU), donde se realiza todo el trabajo, y la memoria, donde se almacenan todos los programas y datos.

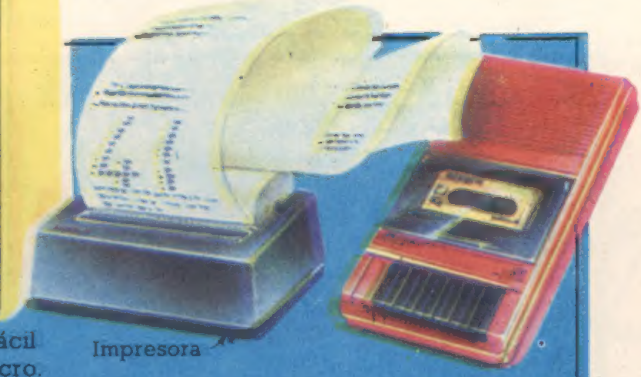




En realidad, la computadora tiene dos memorias. Una, llamada ROM, contiene el programa que controla todas las operaciones de la computadora. La otra, llamada RAM, es una memoria vacía donde tú almacenas todos tus programas y datos. Cuando desconectes un micro, toda la información en el RAM se borra, pero los programas en el ROM permanecen.



Una pantalla de TV es la forma más fácil de presentar la información de un micro. También se puede imprimir en papel, utilizando una impresora, lo que es muy útil, ya que la información que aparece en la pantalla desaparecerá al apagar ambos aparatos.



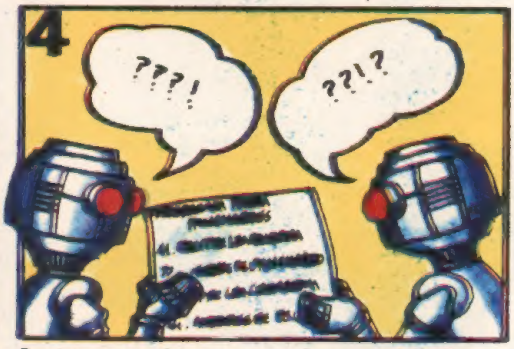
Impresora

Otra forma de almacenar información de un micro es grabarla en una cinta magnetofónica. Se puede grabar y volver a colocar en el micro cuando se necesite de nuevo.

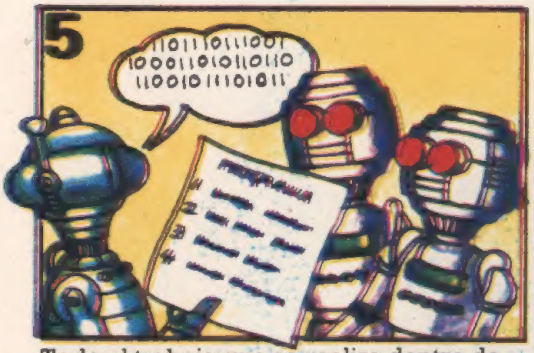
Dar instrucciones a la computadora

Para lograr que la computadora haga algo tienes que darla una instrucción que entienda. Debe ser una orden directa que pueda seguir o un programa de instrucciones que guarde en su memoria y no se ponga en marcha hasta que tú le des la orden de adelante.

Las instrucciones de un programa deben ser cuidadosamente elaboradas. La computadora las llevará adelante con exactitud, incluso aunque fuesen erróneas.



La computadora no puede comprender las instrucciones dadas en nuestro lenguaje; es decir, hay que describirlas en uno de los muchos lenguajes de computadora. Algunos de estos lenguajes aparecen en la página siguiente.



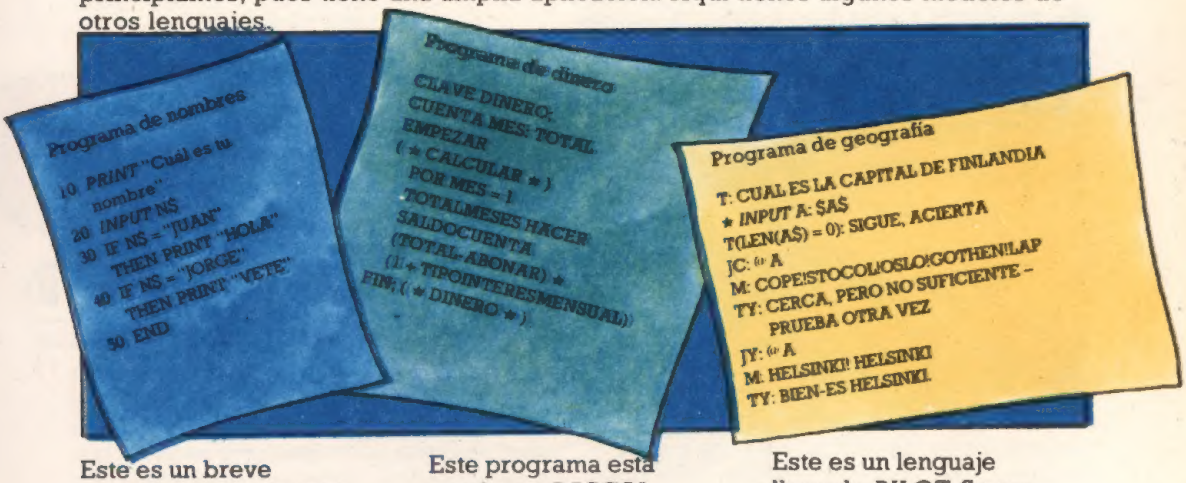
Todo el trabajo que se realiza dentro de una computadora se hace con un código de pequeños impulsos eléctricos. Tus instrucciones se traducen a un código por un programa interno que se llama intérprete o traductor.

Cada pequeña información, en el código de la computadora, se representa por modelos de impulsos. El código puede escribirse utilizando 1 para representar un impulso y 0 para mostrar que no hay impulso.

Lenguajes de computadora

Tú podrías escribir programas en un código de una computadora, pero es muy difícil. Por ello, hay lenguajes especiales de computadoras, llamados lenguajes de alto nivel, que ésta puede traducir a su propio código.

Hay cientos de estos lenguajes, muchos diseñados para hacer trabajos específicos. *BASIC* es uno de los más conocidos, y su nombre está formado por las letras iniciales de Beginner's All-purpose Symbolic Instruction Code (Código de instrucción simbólica para principiantes con objetivos variados). Sin embargo, no es sólo para principiantes, pues tiene una amplia aplicación. Aquí tienes algunos modelos de otros lenguajes.



Este es un breve programa en *BASIC*. El paso 20 dice a la computadora que imprima (print) en la pantalla "cuál es tu nombre". La computadora almacena la respuesta en su memoria, y si su nombre es Juan o Jorge, le da un mensaje.

Este programa está escrito en *PASCAL*, llamado así en recuerdo del famoso matemático francés. Es especialmente interesante para utilizarlo en trabajos de tipo económico. Mucha gente cree que es mejor emplear *PASCAL* que *BASIC*.

Este es un lenguaje llamado *PILOT*. Se usa para escribir programas que ayuden a otras personas a aprender nuevas materias. En este lenguaje, la computadora puede reconocer las respuestas.

11. Ab3, Ce5
12. 0-0-0, Cc4
13. Axc4, Txc4
14. h5, Cxh5

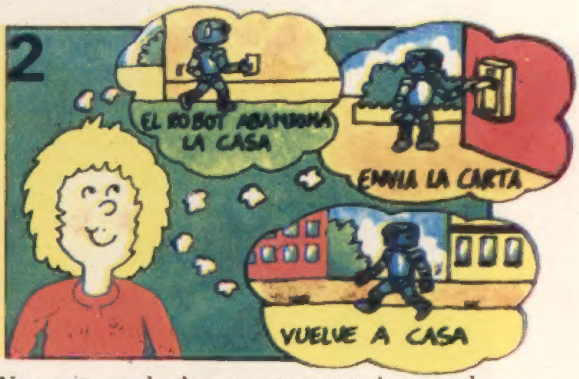
A primera vista, los lenguajes de la computadora parecen extraños y difíciles, como la frase del finlandés que aparece más arriba, pero luego se hacen asequibles según se avanza. Hay muchas materias que utilizan lenguajes especiales. Por ejemplo, en matemáticas se utilizan notaciones especiales para escribir ideas y fórmulas que necesitarían muchas palabras ordinarias para explicarlas y otras clases de notaciones se usan para describir movimientos de ajedrez o notas musicales.

Escribir programas

Un programa es como las reglas de un juego o como un recipiente para un pastel. Si hay un error en las reglas, o en el recipiente, no serás capaz de jugar correctamente o de hacer un pastel bien. De la misma manera, los resultados que obtienes de una computadora dependen de las instrucciones que le des. Para escribir un programa primero necesitarás estudiar cuidadosamente lo que quieres y calcular el número de pasos que necesitas para obtenerlo.

Programa para enviar una carta

Imagina que tienes que escribir un programa para decir a un robot que mande una carta. Una simple instrucción como la de arriba sería muy difícil que el cerebro de la computadora la entendiese.



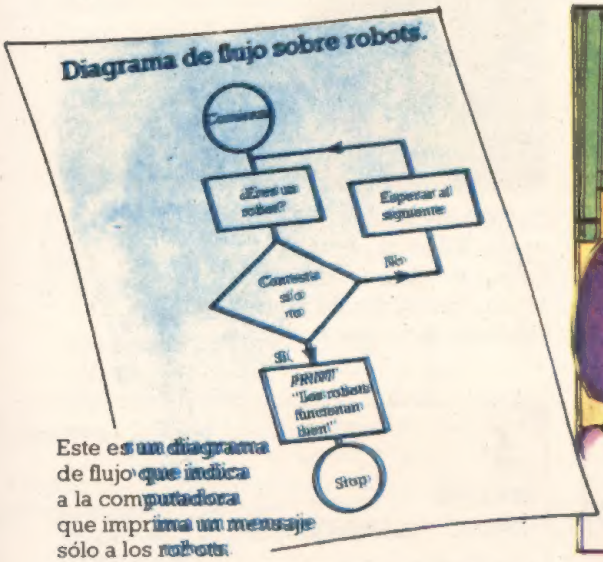
Necesitas calcular exactamente lo que el robot necesita hacer para enviar la carta. Su computadora necesita instrucciones que le digan qué hacer en cada momento.

Para escribir el programa necesitas desmenuzar las instrucciones de cada paso en pasos todavía más sencillos que se puedan traducir al lenguaje que pueda comprender el robot.

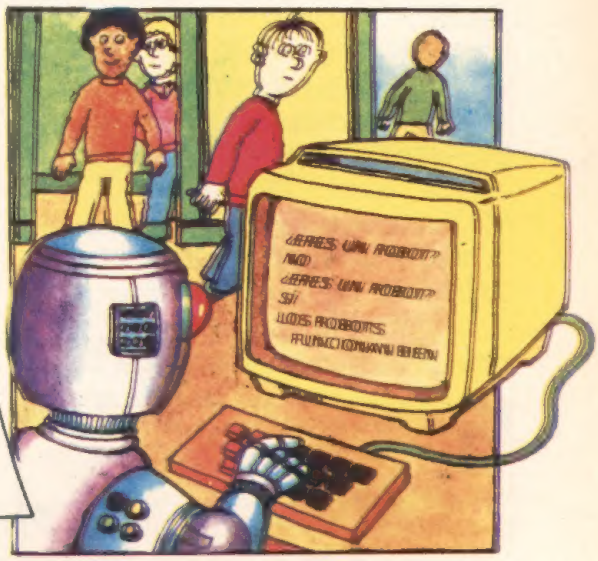
El robot intentará seguir tus instrucciones aunque estén mal o sean incompletas. Los errores llevan a resultados sorprendentes de la computadora.

Diagramas de programas

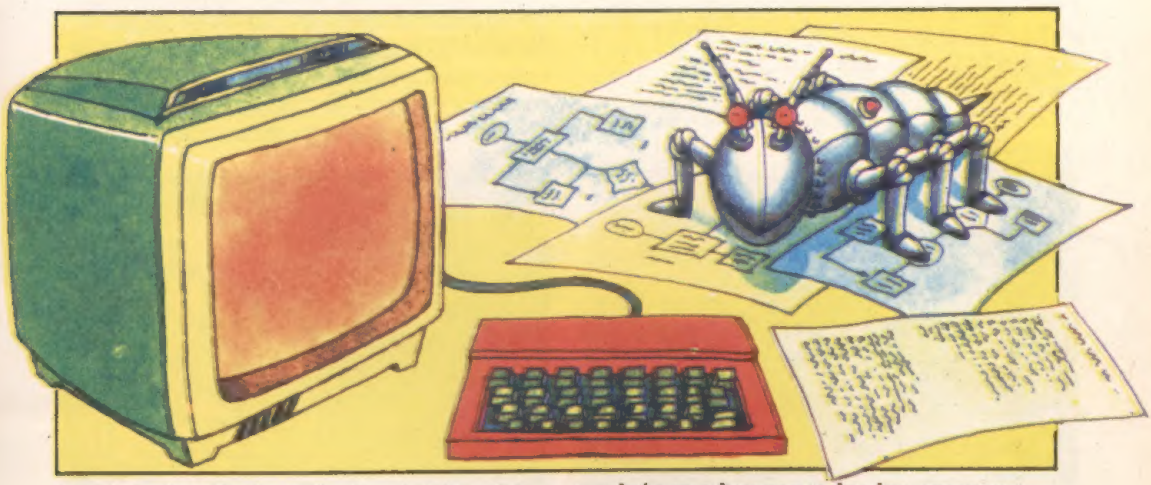
Al escribir un programa suele ayudar el dibujar un diagrama como el inferior, señalando los pasos principales que necesitas para resolver el programa. Un diagrama como éste se denomina diagrama de flujo. Muestra cada uno de los pasos que necesita realizar la computadora y el orden en que deben ir.



Este es un diagrama de flujo que indica a la computadora que imprima un mensaje sólo a los robots.



Un diagrama del flujo tiene encuadres con diferentes formas para cada paso del programa. El principio y el final del programa están en círculos, las instrucciones que se dicen a la computadora están en rectángulos y las decisiones, donde la computadora realizará diferentes cosas dependiendo de la información que la llegue, en encuadres con forma de diamantes. Las líneas muestran las posibles rutas que puede tomar la computadora.



Después de calcular todos los detalles del programa puedes traducirlo a *BASIC* y probarlo en la computadora. Posiblemente, el programa no funcione a la primera, ya que habrá errores. Estos pueden ser de ortografía al escribirlo o bien errores de lógica. Antes de conseguir que funcione tendrás que buscar todos los errores y corregirlos. * En ocasiones, un error provoca un resultado ligeramente diferente que quizá prefieras. Los errores útiles como éste se denominan *PUGS*.

* Encontrarás algunos trucos para buscar errores en las páginas 42-43.

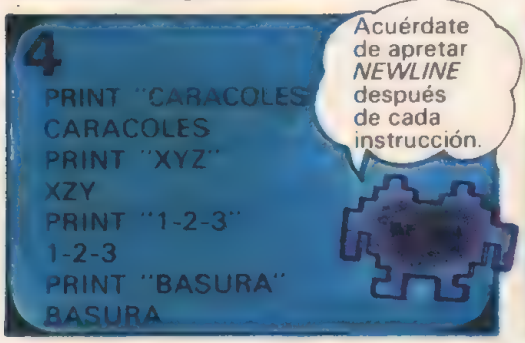
Primeras palabras en BASIC

Muchas palabras del *BASIC* están basadas en palabras inglesas, lo que facilita el adivinar lo que significan. Por ejemplo, *PRINT* significa "imprimir en la pantalla", *RUN* significa "ejecutar el programa" e *INPUT* significa "dar información a la computadora". En estas dos páginas verás como usar la palabra *PRINT*.
Muchas computadoras caseras tienen ya un traductor de *BASIC* en su interior, por lo que cuando las conectas ya están listas para ser programadas en *BASIC*. *

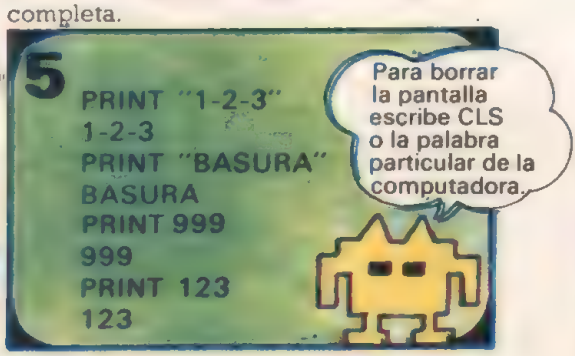
Cuando enciendes una micro normalmente aparecen en la pantalla algunas palabras, junto a un símbolo denominado cursor. El cursor indica dónde aparecerá la siguiente letra que escribas.

Para indicar a la computadora que escriba palabras en la pantalla utiliza *PRINT* con las palabras que quieras entre comillas. Por ejemplo, *PRINT "CARACOLES"* le dices que escriba caracoles en la pantalla.

La computadora no llevará a cabo tus instrucciones hasta que aprietes la tecla *NEWLINE* (o *RETURN* o *ENTER*, según la computadora); es decir, que quieras un paso nuevo, para lo cual le indicas a la computadora que esta instrucción está completa.



La computadora visualizará en pantalla cualquier cosa que escribas entre palabras o símbolos. Observa que no escribe las comillas.



Para escribir números no son necesarias las comillas. Para borrar la pantalla, normalmente escribe CLS. Si tienes computadora, comprueba si ésta es la palabra que utiliza.

* Algunas computadoras necesitan que se les introduzca un programa especial antes de entender el *BASIC*

Un programa en *BASIC*

En un programa, cada instrucción comienza con un número. Esto indica a la computadora que debe almacenar las instrucciones en la memoria, sin llevarlo a cabo hasta que tú se lo ordenes. En la página opuesta, las instrucciones para la computadora no tenían números, por lo que la computadora las realizaba inmediatamente. Este es un pequeño programa en el que la computadora representa en la pantalla una cara basándose en símbolos.

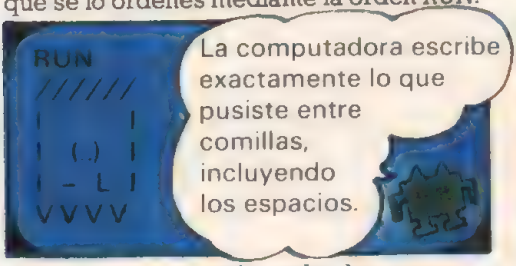
En algunas computadoras el signo O tiene una raya atravesada como ésta.


```
10 PRINT "//////"
20 PRINT "I    I"
30 PRINT "I (..) I"
40 PRINT "I -L- I"
50 PRINT "VVVV"
60 END
```

Los números de los pasos suelen ir de diez en diez para que puedas añadir instrucciones entre medias sin necesidad de cambiar toda la numeración.

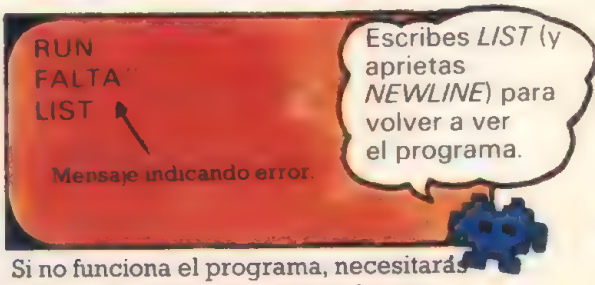
Muchas computadoras no necesitan este paso

Cuando introduces un programa has de apretar la tecla *NEWLINE* (o la que necesita la computadora) al final de cada paso. Los pasos quedan escritos en la pantalla, pero no se llevan a cabo hasta que se lo ordenes mediante la orden *RUN*.



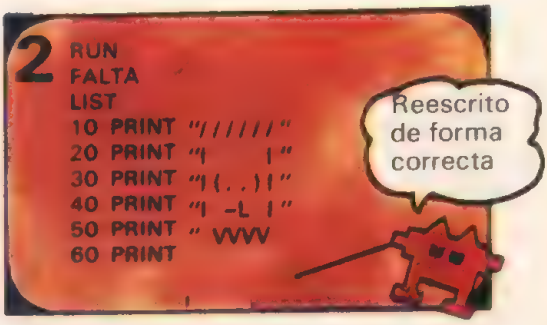
Cuando hayas escrito todos los pasos, revísalos para no tener errores. Luego indícale a la computadora que ejecute el programa, escribiendo *RUN* y apretando después *NEWLINE*.

Ten cuidado de no confundir la letra *O* con el cero numérico, ya que producirás múltiples errores. Muchas computadoras tienen un orden para borrar y corregir errores: *RUBOUT* o *DELETE*.



Si no funciona el programa, necesitarás volver a verlo para localizar el error. Para conseguir esto escribes *LIST*. En ocasiones la computadora te dará un mensaje diciendo dónde está el error.

Para casi todos los errores la computadora te dará un mensaje. Los mensajes de errores se explican en el manual de la computadora. La manera más fácil de corregir un error es volver a copiar todo el paso. La computadora sustituirá el paso antiguo por el nuevo. Para librarte

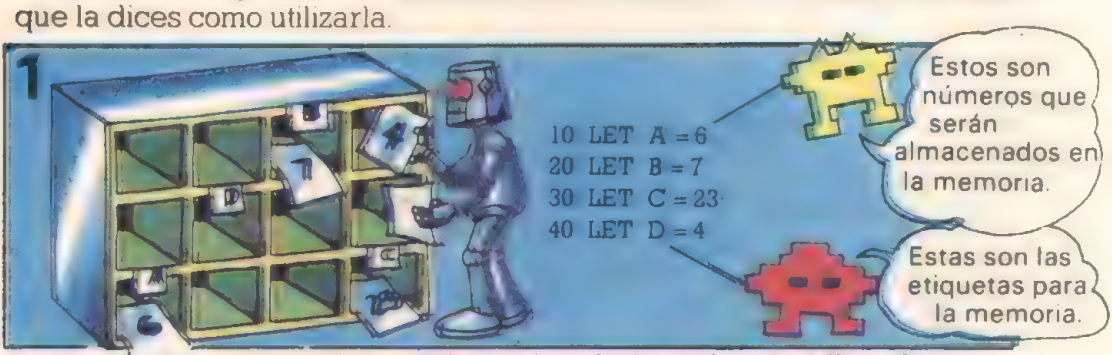


totalmente de un paso, simplemente escribe el número del paso y aprieta *NEWLINE*. Cada computadora tiene incluso su propia manera de corregir o alterar partes de pasos, utilizando palabras como *EDIT* o *COPY*. Esto se explica en el manual de la computadora.

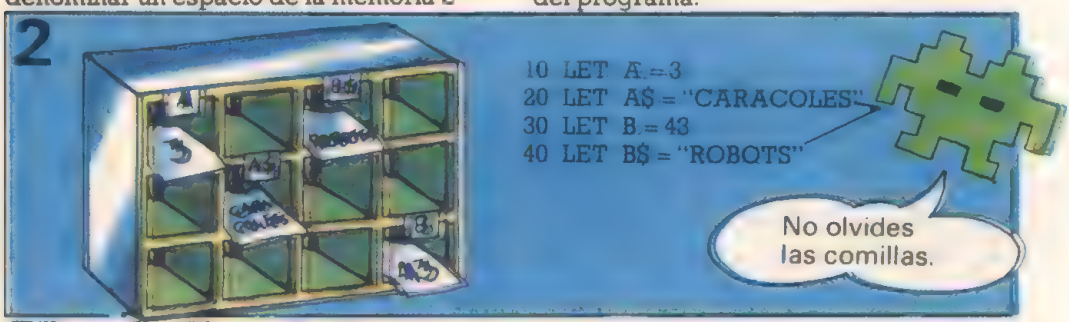
Ejercicio.—Prueba a cambiar el programa para alterar los rasgos de la cara.

Dar información en la computadora

Para que la computadora haga algo más útil que simplemente representar cosas en la pantalla, tienes que darle información o *"data"* con la que pueda trabajar. La computadora almacena esta información en su memoria hasta que la dices como utilizarla.

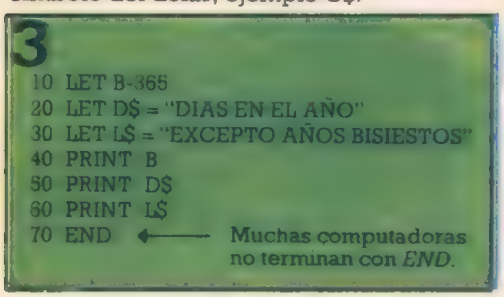


Cuando introduces datos en la memoria de la computadora has de darles un nombre para que puedas volver a encontrarlos. Puedes utilizar las letras del alfabeto como etiquetas. Para denominar un espacio de la memoria e introducir un número en él puedes utilizar la palabra *LET* tal y como se indica arriba. Un espacio etiquetado se denomina una variable, ya que puede contener datos diferentes en cada etapa del programa.



Utiliza un tipo diferente de etiqueta para almacenar letras y símbolos en los espacios de la memoria. Las letras y los símbolos se denominan *"strings"*, es decir, series de caracteres, y por ellos se utilizan las letras del alfabeto con el símbolo del dólar, ejemplo C$. *

Introduces una serie de caracteres en un espacio de la memoria mediante el *LET*, de la misma manera que para una variable numérica, pero las letras y los símbolos deben estar entre comillas tal y como se indica en el ejemplo superior.



Para exhibir la información en la pantalla se utiliza la palabra *PRINT* con el nombre de la variable. Ej.: PRINT A$. Este corto programa escribe la información de las variables B, D$ y L$.

Puedes ejecutar el programa tantas veces como quieras. La computadora siempre ofrecerá la misma información. Los datos en las variables se conservan hasta que los cambies.

* Se conoce con "C dólar" o "C strig».

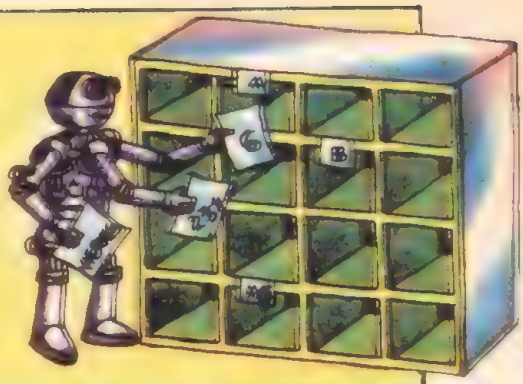
Otro método

```
10 READ A
20 READ B
30 READ A$
40 DATA 6, 231, VIERNES
```

Debes colocar el tipo correcto de etiqueta para números y letras.

Comas

Algunas computadoras necesitan la palabra *DATA* entre comillas.



Otra manera de almacenar información es mediante *READ* Y *DATA*. Los pasos *READ* indican a la computadora que etiquete espacios de la memoria, estando la información contenida en el *DATA*.

Algunos programas

1

```
10 READ Q
20 READ X$
30 DATA 24, PAN FRIO
40 PRINT Q
50 PRINT X$
60 END
RUN
24
PAN FRIO
```

Coma

Esto es una sección de información, incluyendo el espacio.

Aquí hay dos pogramas, uno utilizando *DATA* y otro utilizando *LET*, ambos para almacenar información en la memoria de la computadora.

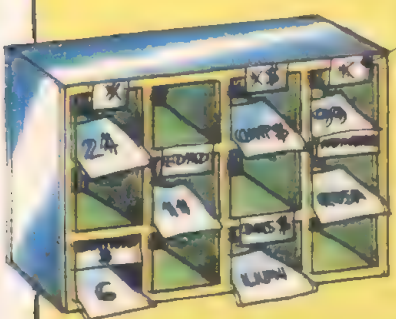
Cuando ejecutes el programa, la computadora coloca cada sección de información en un espacio de la memoria. Las secciones de información han de estar separadas por comas para que la computadora pueda diferenciarlas. *

2

```
10 LET A$ = "LOS ROBOTS SON
            MAGNIFICOS"
20 LET B$ = "SI PREFIERES"
30 LET C$ = "SON GRANDISIMOS
            IDIOTAS DE METAL"
40 PRINT A$
50 PRINT B$
60 PRINT C$
70 END
RUN
LOS ROBOTS SON MAGNIFICOS
SI PREFIERES
SON GRANDISIMOS IDIOTAS
DE METAL
```

Comillas

Más sobre variables



Variable numérica

No puedes utilizar estas palabras como nombres de variable, ya que no contienen palabras *BASIC*.

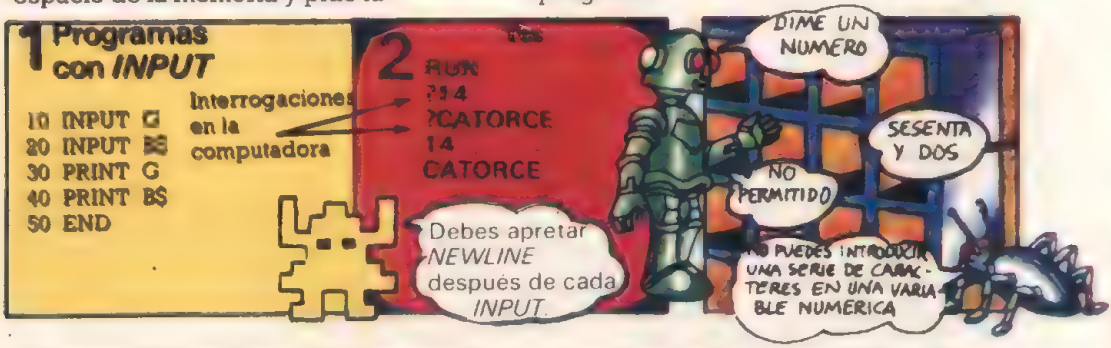
Las variables son espacios etiquetados de la memoria en los que se almacena la información. Una variable que contenga números se denomina variable numérica, y una que contenga letras y símbolos se denomina variable en cadena. El contenido de las variables puede cambiar mientras se ejecute el programa. Algunas computadoras pueden utilizar palabras como nombres de variables, siempre que estas palabras no contengan palabras *BASIC*, ya que esto confundiría a la computadora.

* No puedes utilizar este método en la computadora ZX81.

Utilizar INPUT

Otra manera de introducir información en la computadora es mediante la palabra *INPUT*. Esta te permite introducir información mientras se ejecuta el programa, pudiendo utilizarse diferente información cada vez que ejecutes el programa.

Utilizar *INPUT con una etiqueta como A para números y A$ para series de caracteres. Cuando la computadora encuentra la palabra INPUT* en un programa coloca la etiqueta en un espacio de la memoria y pide la información, normalmente escribiendo una interrogación, u otro símbolo, en la pantalla. Luego escribes tú la información y la computadora almacena en el espacio de la memoria y continúa con el resto del programa.



El segundo dibujo muestra lo que sucede cuando ejecutas este programa. Cuando la computadora encuentra la palabra *INPUT* en la línea 10, escribe una interrogación en la pantalla y espera a que escribas un número para G. Luego escribe otra interrogación para la instrucción *INPUT* en la línea 40. Esta vez has de introducir palabras o símbolos, ya que la variable B$ hace que la computadora espere una serie de caracteres.

Si tienes una computadora, prueba a introducir este programa, luego aprieta *RUN* para empezarlo. Cuando la computadora te pida información, escribe tu nombre y edad o cualquier nombre ridículo y una cifra, como el ejemplo superior. Ejecútala varias veces con diferentes datos, apretando *RUN* para empezar el programa cada vez. La computadora siempre escribe lo que tu introduzcas en N$ y A.

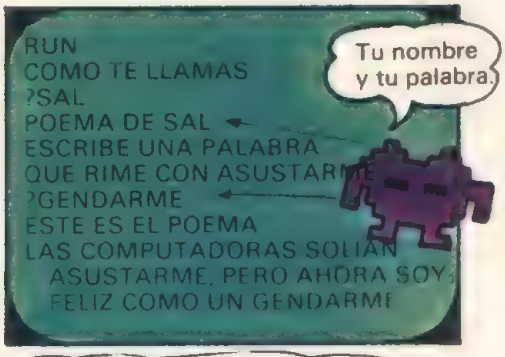
Escribir programas poéticos

Ahora sabes suficiente *BASIC* para escribir un poema con la computadora. Este es un programa para escribir poesía que utiliza *PRINT* e *INPUT*.

```
10 PRINT "COMO TE LLAMAS"
20 INPUT N$
30 PRINT "POEMA DE"
40 PRINT N$
50 PRINT "ESCRIBE UNA PALABRA"
60 PRINT "QUE RIME CON ASUSTARME"
70 INPUT A$
80 PRINT "ESTE ES EL POEMA"
90 PRINT "LAS COMPUTADORAS
   SOLIAN ASUSTARME"
100 PRINT "PERO AHORA SOY FELIZ
    COMO UN"
110 PRINT A$
120 END
```

Este paso escribe tus palabras.

Este paso escribe tu nombre.



Este programa hace que la computadora pregunte tu nombre, lo almacena en N$ y lo escribe en el paso 40. Almacena la palabra que escoges en A$ y la escribe como parte del programa en el paso 110.

Si tienes una computadora, prueba a ejecutar el programa varias veces, introduciendo distintas palabras en el paso 70.

Ejercicio

¿Puedes escribir un programa mediante el cual la computadora te pregunte tu nombre y luego escriba hola, seguido de tu nombre y un mensaje para ti?

Instrucciones para introducir un programa

1. Antes de empezar un programa escribe *NEW*. Esto borra cualquier programa anterior, al igual que cualquier variable que haya en la memoria de la computadora.
2. Mientras escribes el programa, acuérdate de apretar *NEWLINE* o cualquiera que sea la palabra de tu computadora al final de cada paso.
3. Después de escribir el programa, comprueba todos los pasos en la pantalla para no cometer errores de sintaxis. Asegúrate también de que no te has olvidado ningún paso.
4. Después puedes escribir CLS (o la palabra de tu computadora) para borrar el programa de la pantalla. Aprieta luego *RUN* para ejecutar el programa.
5. Para obtener el listado del programa, para comprobarlo o cambiar algún paso, escribe *LIST*. Para ver un determinado paso puedes escribir *LIST* con el número del paso, debes comprobar esta instrucción, ya que suele cambiar en cada computadora.
6. Para el programa mientras está siendo ejecutado, escribe *BREAK* o *ESCAPE*. También debes comprobar esta instrucción, ya que varía en algunas computadoras. En algunas computadoras *ESCAPE* borra todo el programa de la memoria. Para volver a iniciar el programa escribe *RUN*.

En las págs. 42 y 43 encontrarás trucos para descubrir errores.

Hacer cosas con PRINT

Hasta ahora has visto cómo utilizar *PRINT* para escribir palabras y números en la pantalla, y para mostrar el contenido de las variables. Debajo aprenderás como usar comas y punto y coma para dejar espacio entre los datos de la pantalla.

También puedes utilizar *PRINT* para hacer cálculos en la computadora. Al pie de la página verás cómo. En la página opuesta verás más cosas para hacer con variables.

Comas y punto y coma

```
10 PRINT "ESTOY SEPARADO",
20 PRINT "UN POCO"
```
Coma

ESTOY SEPARADO UN POCO

```
10 PRINT "ESTOY TOTALMENTE";
20 PRINT "COMPRIMIDO"
```
Punto y coma

ESTOY TOTALMENTE COMPRIMIDO

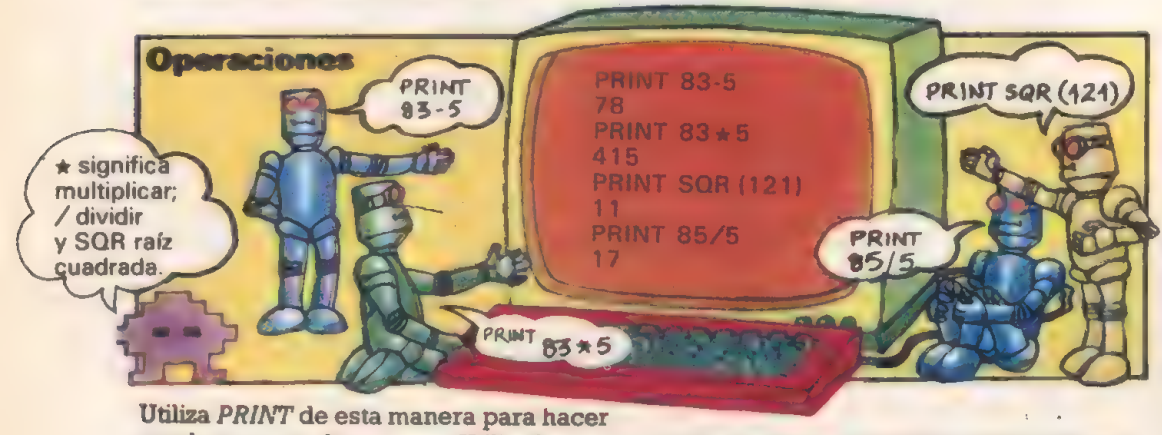
```
10 PRINT "ESTOY REALMENTE"
20 PRINT
30 PRINT "SEPARADO"
```

ESTOY REALMENTE

SEPARADO

La palabra *PRINT* en solitario produce esta línea vacía.

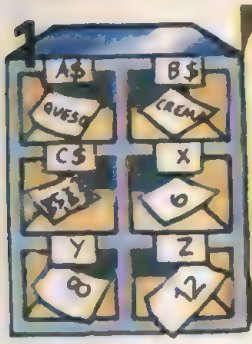
Estos pasos muestran cómo utilizar coma y punto y coma para indicar a la computadora dónde escribir la siguiente línea. Una coma indica que la coloque a distancia y un punto y coma que la coloque donde está. El recuadro superior muestra cómo aparecerían los textos en la pantalla; la palabra *PRINT* por sí sola indica a la computadora que deje una línea vacía.



Utiliza *PRINT* de esta manera para hacer que la computadora opere. Utiliza los signos habituales para sumar y restar; para multiplicar y/ para dividir.

La computadora también puede realizar cálculos matemáticos complicados como senos, cosenos, raíces cuadradas, etc.

Más sobre variables



2

En casi todas las computadoras necesitas dejar un espacio a ambos lados de la variable, dentro de las comillas.

Espacios

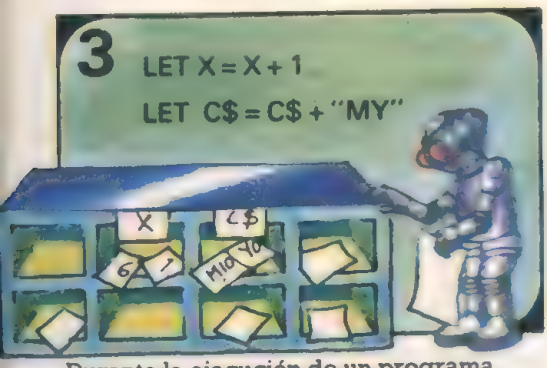
```
PRINT " COMI "; X; " SANDWICHES DE "; A$; " Y PASTA DE CACAHUETES "
COMI 6 SANDWICHES DE QUESO Y PASTA DE CACAHUETES

PRINT " COMI "; Z; " SANDWICHES DE "; C$; " Y PASTA DE CACAHUETES "
COMI 12 SANDWICHES DE MERMELADA Y PASTA DE CACAHUETES
```

Escribir variables solas no puede ser demasiado útil. Normalmente necesitarás algunas palabras junto a ellas para saber lo que significan. Para escribir variables y palabras juntas, las palabras deben ir entre comillas y la variable debe tener un punto y coma a cada lado, tal y como se muestra en el recuadro superior. Si quieres separar los datos, utiliza comas en lugar de punto y coma.

3

```
LET X = X + 1
LET C$ = C$ + "MY"
```



Durante la ejecución de un programa puedes cambiar los contenidos de los espacios de la memoria de esta manera. Para la computadora estas órdenes significan sumar uno a la cifra que ocupa el espacio X de la memoria y "MY" añadirlo a las letras en C$.

4

Espacios

```
PRINT " COMI "; X; " SANDWICHES
DE "; C$; " Y PASTA DE CACAHUETES

COMI 7 SANDWICHES DE MERMELADA
Y PASTA DE CACAHUETES
```

La próxima vez que ordenes a la computadora que escriba las variables, mostrará las nuevas palabras y cifras almacenadas en los espacios de la memoria.

5

```
10 LET A = 9
20 LET B = 7
30 PRINT A * B
40 PRINT A/B
50 END
RUN
63
1.28571
```

Multiplicar

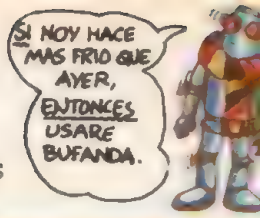
Dividir

También puedes hacer sumas con variables, tal y como se indica en el programa superior. La computadora encuentra las cifras en los espacios de la memoria y realiza las operaciones.

Ejercicio

1. Escribe un programa para sumar números a las variables en el programa de la izquierda, de manera que escriba como solución 100 y 1 en un mismo renglón con un espacio entremedias.
2. Cambia los pasos 30 y 40 para que escriban los números, lo que estás haciendo con ellos y la solución. Por ejemplo: "7 veces 9 es 63"
3. Cambia tu respuesta al puzzle de la página 15 para que escriba tu nombre y el mensaje en un mismo renglón.

¿Cómo comparan las computadoras?



Una de las cosas más útiles que puede hacer una computadora es comparar datos y realizar cosas diferentes según los resultados que obtenga.

La computadora puede realizar diferentes pruebas con la información para compararla. Los símbolos para las pruebas los verás arriba. Puede comparar para saber si dos datos son iguales, diferentes, si uno es mayor o menor que el otro.

2

```
IF A = B THEN PRINT "SON IGUALES"
IF A   . THEN PRINT "A ES MAYOR"
IF A   THEN PRINT "A ES MENOR"
IFIA     B THEN PRINT "NO SON
IGUAL"
```

Estos pasos muestran cómo utilizar IF y THEN para hacer que la computadora compare los datos. Puedes comparar cualquier tipo de información —palabras, cifras y variables—; es decir, los contenidos de los espacios de la memoria.

3 Programa sobre el tiempo

```
10 PRINT "COMO ESTA EL TIEMPO HOY"
20 INPUT W$
30 IF W$ = "LLUEVE" THEN PRINT
   "TIEMPO DE PARAGUAS"
40 IF W$ = "SOLEADO" THEN PRINT
   "ESTUPENDO"
50 END
```

Aquí hay un programa en el que se utiliza IF y THEN. En el paso 20, la computadora almacena la palabra que introduces en la variable W$. Luego, en los pasos 30 y 40, comprueba si la palabra W$ es igual a "llueve" o "soleado". Si lo es, escribirá una de las respuestas. Si introduces una palabra diferente en el paso 20 no pasará nada. Puedes cambiar las palabras de los pasos 30 y 40; si lo haces, introduce en *INPUT* alguna de las nuevas palabras.

4

```
RUN
COMO ESTA EL TIEMPO
HOY
?SOLEADO
ESTUPENDO
RUN
COMO ESTA EL TIEMPO
HOY
?LLUEVE
TIEMPO DE PARAGUAS
```

5 Programa de edad

```
10 PRINT "QUE EDAD TIENES"
20 INPUT A
30 IF A>16 THEN PRINT "VIEJO"
40 IF A<16 THEN PRINT "JOVEN"
50 IF A=16 THEN PRINT
   "EDAD JUSTA"
RUN
QUE EDAD TIENES
?16
EDAD JUSTA
```

En el programa de la edad, la computadora compara *INPUT* A con la cifra 16. Si es mayor que 16, aparecerá "viejo". Si es menor, aparecerá "joven", y si es 16, aparecerá "edad justa". En el otro programa, la computadora escribe una de las dos posibilidades según si A$ es igual a "rouge" o no.

6 Lección de francés

```
10 PRINT "COMO SE DICE ROJO EN
   FRANCES"
20 INPUT A$
30 IF A$="ROUGE" THEN PRINT
   "CORRECTO"
40 IF A$<>"ROUGE" THEN PRINT
   "NO, ROUGE"
RUN
COMO SE DICE ROJO EN FRANCES
?BIEN
NO, ROUGE
```


Ejercicio.—¿Puedes escribir un programa para que la computadora te haga sumar, y al poner la solución te diga "correcto", o te dé la respuesta correcta?

Programas ramificados

1

```
IF A=6 THEN LET A$="SEIS"
IF X=Y-2 THEN LET Z=0
IF S=T THEN STOP
IF R<10 THEN GOTO 30
```

Esto hace que la computadora salte hasta el paso 30

2

```
10 PRINT K$
20 IF K$="SI" THEN GOTO 100
30 IF K$="NO" THEN GOTO 200

100 PRINT "ESCRIBISTE SI"
110 STOP

200 PRINT "ESCRIBISTE NO"
210 END
```

Estos dos pasos hacen que la computadora se ramifique a otras partes.

Puedes dar a la computadora casi cualquier instrucción detrás de la palabra *THEN*, tal y como se indica en el recuadro superior. Una instrucción útil es hacer que vaya a otro paso. (En casi todas las computadoras, no la ZX81, puedes omitir la palabra GOTO.) Normalmente necesitas una instrucción STOP en los programas con GOTO, o la computadora irá repitiendo el programa sin acabar nunca.

Programa de matemáticas

```
10 PRINT "ESCRIBE UN NUMERO"
20 INPUT A
30 PRINT "ESCRIBE OTRO NUMERO"
40 INPUT B
50 PRINT "QUIERES"
60 PRINT "SUMAR, RESTAR, MULTIPLICAR"
65 PRINT "DIVIDIR O STOP"
70 INPUT C$
80 IF C$="SUMAR" THEN PRINT A+B
90 IF C$="RESTAR" THEN PRINT A-B
100 IF C$="MULTIPLICAR" THEN PRINT A*B
110 IF C$="DIVIDIR" THEN PRINT A/B
120 IF C$="STOP" THEN STOP
130 GOTO 10
```

```
RUN
ESCRIBE UN NUMERO
?17
ESCRIBE OTRO NUMERO
?185
QUIERES
SUMAR, RESTAR, MULTIPLICAR
DIVIDIR O STOP
?SUMAR
201 <-- Respuesta de la computadora
ESCRIBE UN NUMERO
?
```

EL PROGRAMA SOLO SE PARARÁ CUANDO ESCRIBAS LA PALABRA STOP

En este programa los números se almacenan en A y B, y tus instrucciones se almacenan en C$. En los pasos 80 a 120 la computadora compara C$ con cinco palabras diferentes, y cuando encuentra la palabra adecuada lleva a cabo la instrucción.

Programa para adivinar la edad

1

```
10 PRINT "ADIVINA MI
   EDAD
30 IF G < > 14 THEN PRINT
   "VUELVE A PROBAR"
40 IF G < >14 GOTO 20
50 PRINT "CORRECTO"
60 END
```

2

```
RUN
ADIVINA MI EDAD
?15
VUELVE A INTENTARLO
?14
CORRECTO
```

3

```
ADIVINA MI EDAD
?15
MAS JOVEN
?13
MAYOR
14
CORRECTO
```

¿PUEDES ESCRIBIR UN PROGRAMA QUE HAGA ESTO?

Este programa irá repitiéndose hasta G=14. Cuando G=14 la computadora se saltará los pasos 30 y 40 y escribirá "correcto". ¿Puedes variar el programa para que te dé pistas, tal y como se indica en el recuadro superior?

Programas con mucho BASIC

Los programas en estas dos páginas utilizan el *BASIC* que hemos aprendido. El primer programa es un juego espacial para dos personas para jugar con la computadora. Si no tienes una computadora, estudia los programas e intenta ver cómo funcionan.

Comando espacial

```
10 PRINT "HORIZONTALES DEL INVASOR"
20 INPUT A
30 PRINT "VERTICALES DEL INVASOR"
40 INPUT B
50 CLS
60 PRINT "HORIZONTALES DEL COMANDO"
70 INPUT C
80 PRINT "VERTICALES DEL COMANDO"
90 INPUT D
100 CLS
110 LET X=SQR ((A-C) * (A-C)+(B-D) * (B-D))
120 PRINT "ESTOS A"
130 130 PRINT X; "ESPACIAL DE DISTANCIA"
140 IF X < 1.6 THEN PRINT "INVASOR LOCALIZADO"
150 IF X < 1.5 THEN STOP
160 PRINT "¿CUAL ES TU POSICION ACTUAL?"
170 GOTO 10
180 END
```

Los pares 10 al 40 almacenan las coordenadas del invasor en A y B.

El paso 50 limpia la pantalla de todos los numeros.

Los pasos 60 a 90 almacenan las coordenadas del comando en C y D.

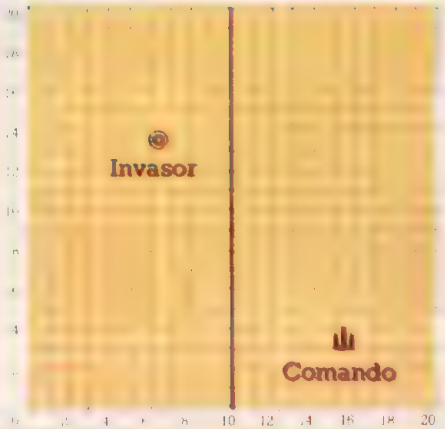
Este paso calcula a qué distancia se hallan y la almacenan en X.

Si X es menor que 1,5, el programa se detiene. Si es menor que 1,5, el programa se repite.

En este juego, una persona hace de invasor y la otra de comando en busca del invasor. Cada jugador dibuja un mapa secreto en el que señala su posición (debajo verás como hacer esto). Dan las coordenadas de sus posiciones y la computadora calcula a qué distancia se hallan. Los jugadores utilizan las cifras de la computadora para calcular su próximo movimiento.

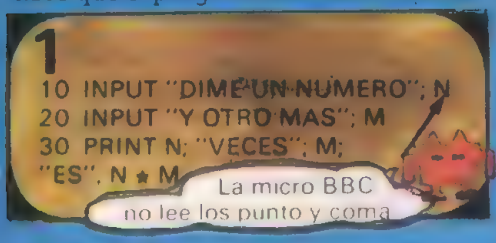
Cómo jugar

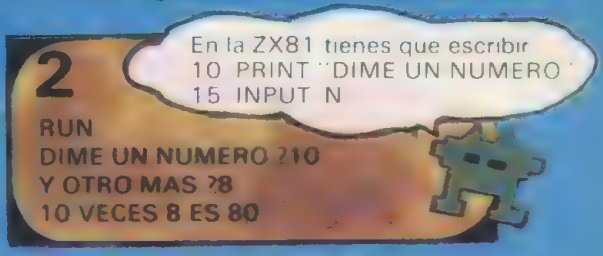
Para el mapa secreto, cada jugador dibuja un mapa cuadriculado de 20 × 20 casillas, colocando los números como se indica a la derecha. El invasor comienza en la parte izquierda y el comando en la derecha. En cada movimiento pueden avanzar dos casillas hacia arriba, hacia abajo, de lado o diagonalmente, introduciendo en la computadora la nueva posición. Cuando están a menos de 1,5 unidades (casillas) de separación, el comando ha captura al invasor.



Cómo hacer que la computadora parezca inteligente

En este programa la computadora parece responder a tus respuestas con sus preguntas. En los dibujos de la parte interior de la página puedes ver cómo funciona el programa. El programa utiliza *INPUT* de una manera ligeramente diferente, lo que hace que el programa sea más cómodo.





En casi todas las computadoras puedes hacer que los pasos con *INPUT* resulten claros utilizando palabras entre comillas antes de la variable.

Cuando ejecutes el programa, la interrogación del *INPUT* aparece detrás de las palabras.

El programa

```
5  LET C = 0
10 PRINT "ME GUSTARIA HABLAR CONTIGO"
20 INPUT "DIME ALGO DIVERTIDO QUE TE
   HAYA PASADO DURANTE ESTA SEMANA"; A$
30 READ B$
40 PRINT B$;
50 INPUT C$
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA POR QUE, PERO POR QUE
95 DATA POR QUE, PUEDES EXPLICARLO
98 DATA PUEDES DECIR POR QUE, CUAL FUE LA RAZON
100 PRINT "ENTONCES LA RAZON QUE ESCRIBISTE"
110 PRINT "          "; A$
120 PRINT "FUE MOTIVADA POR TU RESPUESTA"
130 PRINT "          "; C$
140 PRINT "¡QUE EXTRAÑO!"
150 PRINT "VUELVE A EJECUTARME PARA TENER
    MAS RESPUESTAS BRILLANTES"
160 END
```

Este es el nuevo método de *INPUT*. Tu respuesta se almacena en A$.

En el paso 30 la computadora busca en el primer paso con *DATA* y toma el primer dato colocándolo en B$.

Tu respuesta se almacena en C$.

La variable C en los pasos 60 y 70 es un contador. Lleva la cuenta del número de veces que se repite el programa. Cuando C=6 todos los datos de *DATA* han sido usados y la computadora avanza hasta el paso 100.

El paso 80 hace que la computadora vuelva al paso 30 y sustituya el *DATA* en B$ con el siguiente dato de la lista del *DATA*.

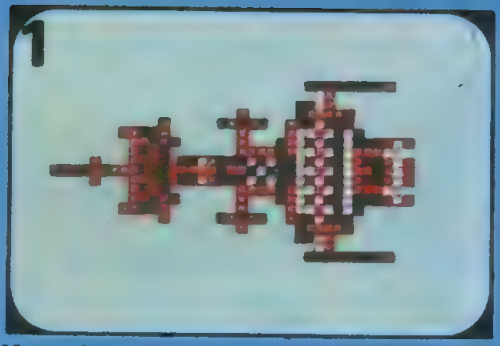
Los espacios en las líneas 110 y 130 dejan espacios en la pantalla delante de tus respuestas. No importa cuántos espacios dejas en el programa.

Cómo funciona

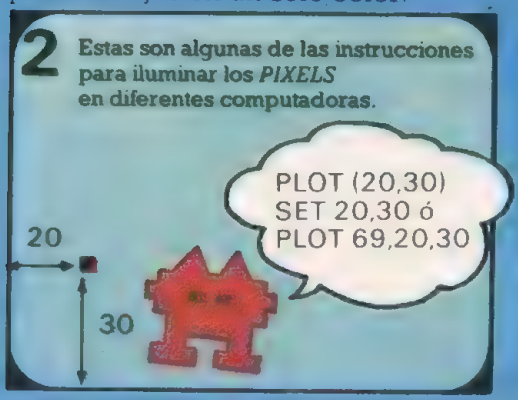
Hacer dibujos

La computadora hace dibujos iluminando pequeños rectángulos de la pantalla. Cada rectángulo se denomina *PIXEL* y necesita una instrucción por separado de la computadora para iluminarse. Muchas computadoras pueden incluso iluminar los *PIXELS* de diferentes colores.

En estas dos páginas puedes aprender a utilizar el *BASIC* para que la computadora realice dibujos en la pantalla. Las instrucciones que se dan aquí son para dibujos en un solo color.

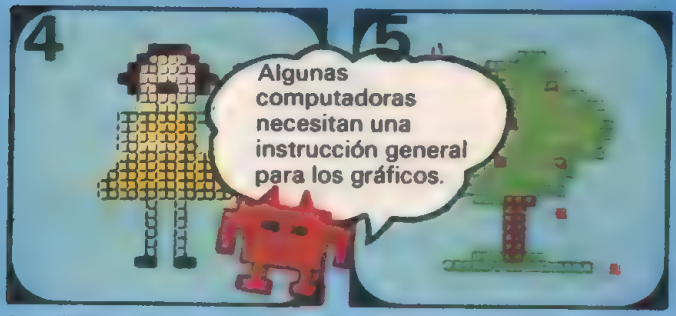


Normalmente puedes ver los *PIXELS* en los dibujos. Una computadora con una amplia memoria hará dibujos con cientos de pequeños *PIXELS*. Estos se denominan gráficos de alta resolución



Las instrucciones para iluminar un *PIXEL* varía en las diferentes computadoras, pero suele ser algo como *PLOT* (X, Y). Siendo X e Y las coordenadas, X el número de *PIXELs* a lo ancho e Y el número hacia arriba.

En computadoras con gráficos de alta resolución puedes iluminar 1000 puntos a lo ancho y otros 1000 de altura. Una computadora menos potente suele tener 60 × 40. (Si tienes una computadora, comprueba el tamaño de la imagen, ya que sino puedes tener errores.)



Los dibujos realizados por la computadora se denominan gráficos. Algunas computadoras necesitan un comando especial antes de realizar los gráficos. Por ejemplo: la micro BBC necesita la palabra *MODE* con un n.º *

Puedes también desconectar un *PIXEL* con un comando como *UNPLOT* (X, Y). En los programas de este libro utilizamos *PLOT* y *UNPLOT*. Si tienes una computadora, comprueba estas instrucciones en el manual.

* Para los programas de este libro utiliza *MODE* en la micro BBC con el comando *PLOT* 69,X,Y. Para borrar usa *PLOT* 71,X,Y.

Representar programas

1

```
10 PRINT "ESCRIBE
   DOS NUMEROS"
20 INPUT X
30 INPUT Y
40 PLOT (X,Y)
50 GOTO 10
```

La instrucción *PLOT* varía en algunas computadoras.

Debes apretar *NEWLINE* o *RETURN* después de introducir cada número.

2

```
RUN
ESCRIBE DOS NUMEROS
?24
?24
ESCRIBE DOS NUMEROS
?30
?15
```

1.er pixel

2.º pixel

Este programa pide dos cifras, colorea el pixel al que correspondan las cifras al actuar de coordenadas. Este programa ha de usar cifras dentro de las posibilidades de tu computadora.

El paso 50 hace que el programa se repita sin acabar, siendo la única manera de detenerlo apretar *BREAK*. Puedes introducirla un contador (ver página 21) para ejecutarlo, digamos, seis veces.

Representar un dibujo

```
10 LET X=10
20 LET Y=10
30 PLOT (X,Y)
40 LET X=X+1
50 LET Y=Y-1
60 IF X   14 THEN GOTO 30

100 LET Y=Y+1
110 LET X=X+1
120 PLOT (X,Y)
130 IF X   20 THEN GOTO 100
```

Esto traza una diagonal hacia abajo.

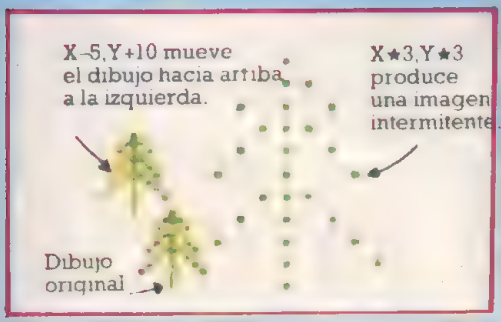
Esto traza una diagonal hacia arriba.

Sumando 1 a X y nada a X se forma una línea horizontal.

Sumando 1 a Y y nada a X se forma una línea vertical.

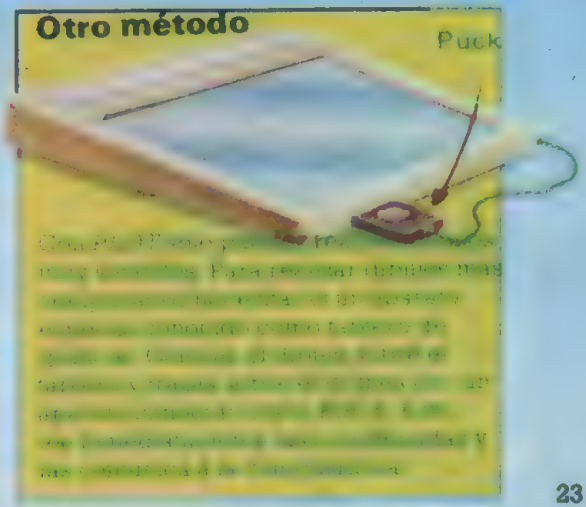
Primero has de representar el dibujo en papel cuadriculado y calcular las coordenadas de los cuadros.

Luego desarrolla el programa para colorear los cuadrados deseados. Dando valores iniciales a X y a Y sumándoles y restándoles, y repitiendo partes del programa, puedes lograr que la computadora coloree secuencias de *PIXELS*, tal y como se indica en la parte superior.

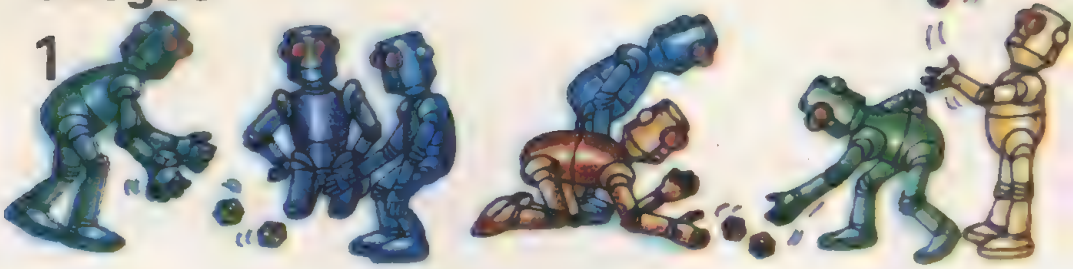


Después de escribir el programa es fácil cambiar el dibujo alterando los números. Puedes cambiarlo de sitio en la pantalla, cambiando los valores iniciales o multiplicando todos los números por tres para producir una imagen "intermitente".

Otro método

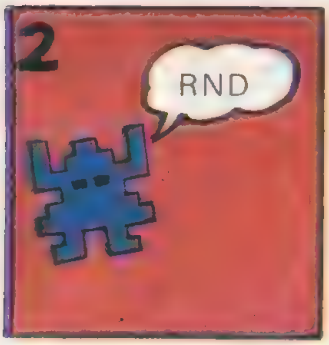


Juegos



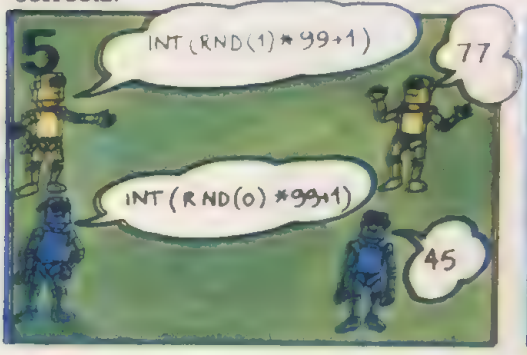
Cuando tiras un par de dados no puedes predecir cuál será el resultado. Las probabilidades son las mismas para cualquier número entre unos y seis. Puedes obtener números impredecibles de una computadora. Se denominan números aleatorios *(random)*.

La computadora contiene un programa especial para producir números aleatorios. En ocasiones repite el mismo número varias veces, pero en consecuencias de muchos números, las veces que sale cada uno suele ser aproximada.



Para hacer que la computadora produzca un número aleatorio se utiliza la palabra clave RND. Algunas computadoras necesitan un 1 o un 0 entre paréntesis. Si tienes una computadora, comprueba en el manual para ver cuál es la instrucción correcta.

La instrucción RND siempre produce un número inferior a uno. En algunas computadoras puedes poner un número entre paréntesis después de RND. Por ejemplo: RND (99). Esto hará que produzca un número entre el 1 y el número entre paréntesis.



En otras computadoras necesitarás la clave INT (diminutivo de *integer*, palabra que significa número entero) seguido de la instrucción RND (ya sea RND (1) o RND (0), dependiendo de la computadora). Luego lo multiplicas por el número que quieras y le sumas uno.

Esta instrucción significa toma un número aleatorio y almacénalo en la variable R. En los programas de este libro utilizaremos INT (RND (1)*60+1) para significar un número entre 1 y 60. Para algunas computadoras necesitarás traducir esta instrucción.

Ejercicio.—¿Saber cómo hacer que la computadora tome un número aleatorio entre 10 y 20?

Ataque espacial

Este es un programa para un juego que utiliza números aleatorios. En el juego te encuentras en un cohete espacial que está siendo atacado por una ola de naves enemigas. El ordenador de tu nave localiza a los enemigos y te da su posición codificada. Para alcanzar a cada enemigo debes hacer funcionar la ignición multiplicando los códigos y escribiendo la respuesta.

```
10 LET C=0
20 LET A=INT(RND(1)*20+1)
30 LET B=INT(RND(1)*20+1)
40 PRINT "CODIGOS DE NAVES EXTRAÑAS"
45 PRINT "SON"; A; B; "DISPARAR"
50 INPUT X
60 LET C=C+1
70 IF X=A*B THEN PRINT "NAVE
   EXTRAÑA DESTRUIDA"
80 IF X <> A*B THEN PRINT "ERRADO"
90 IF C < 6 THEN GOTO 20
100 END
```

C es un contador que cuenta el número de veces que se repite el programa. Cada vez el paso 60 añade 1 a C.

Estos dos pasos generan números aleatorios para los códigos de las naves enemigas y los almacena en A y B.

Su número es almacenado en X.

En los pasos 70 y 80 la computadora comprueba si ha obtenido la respuesta acertada.

Este paso repite el programa si C es menor que 6.

Correr el programa

El cuadro de la derecha muestra lo que ocurre al correr el programa. Si escribes la respuesta correcta para dos números multiplicados, la computadora escribirá "nave enemiga destruida". Si su respuesta no es igual a A×B, la computadora escribirá "errado".

```
RUN
CODIGOS DE NAVES
SON 17      3 DISPARAR
?41
ERRADO
CODIGOS DE NAVES ENEMIGAS
SON 11      5 DISPARAR
?55
NAVE ENEMIGA DESTRUIDA
CODIGOS NAVES ENEMIGAS
SON 13      6 DISPARAR
```

La coma de la línea 45 separa los números de esta forma.

Ejercicio

¿Puedes añadir otro contador a tu programa para contar el número de aciertos e imprimir los resultados al final del programa? Necesitas crear una variable llamada S y darle el valor 0 para empezar, añadiendo al final un 1 por cada acierto.

Programa de modelo aleatorio

```
5 CLS
10 LET X=INT(RND(1)*30+1)
20 LET Y=INT(RND(1)*30+1)
30 PLOT (X,Y)
40 GOTO 10
```

Borra el programa de la pantalla antes que los pixels se iluminen.

Los números aleatorios deben ajustarse a la pantalla de la computadora.

Este paso hace que el programa se repita indefinidamente.

Los comandos para CLS, RND y PLOT pueden variar y algunos necesitan una línea gráfica

Este programa utiliza números aleatorios para encender puntos luminosos (pixels) en la pantalla. Los pasos 10 y 20 crean números aleatorios entre 1 y 30 almacenándolos en X e Y. El paso 30 enciende el pixel que coordina X,Y.

Mientras la pantalla se va llenando puedes ver que aparecen menos pixels, ya que muchos de ellos están ya encendidos. Para detener el programa debes escribir *BREAK* o *ESCAPE* u otra palabra según la computadora.

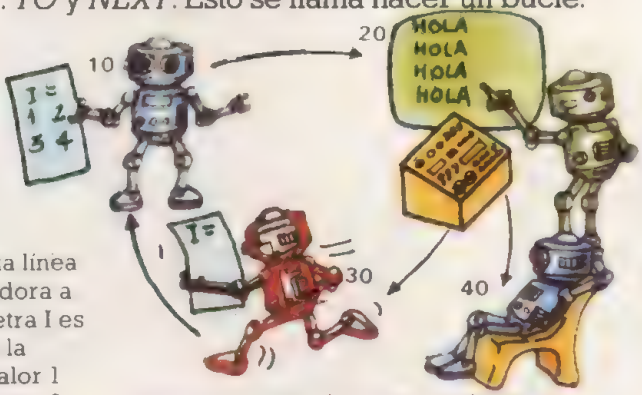
Formación de bucles

A menudo es necesario que la computadora repita la misma cosa varias veces en un programa. En la página 21 puedes ver cómo repetir una parte del programa utilizando el GOTO y una variable que actúa de contador. Otra forma consiste en repetir la misma línea varias veces utilizando las palabras *FOR... TO* y *NEXT*. Esto se llama hacer un bucle.

1 Bucle hola

```
10 FOR I=1 TO 6
20 PRINT "HOLA"
30 NEXT I
40 END
```

Bucle



Este programa tiene un bucle de la línea 10 a la 30 que obliga a la computadora a repetir el paso 20 seis veces. La letra I es una variable y el paso 10 le dice a la computadora que inicie I con el valor 1 en el primer recorrido del programa, 2 en la siguiente, después 3, etc., hasta 6. El paso 20 le dice que escriba la palabra "hola" y el 30 le dice que vuelva para encontrar el siguiente valor de I. Cuando I=6 la computadora va al paso 6.

2 Programa de sumas tontas

```
10 FOR I=10 TO 8
20 PRINT "2 MAS 2 ES 5"
30 NEXT I
40 PRINT
50 PRINT "SOLO BROMEO"
60 END
```

Bucle

Algunas computadoras no tienen signos de exclamación y tú puedes omitirlo.

2 MAS 2 ES 5
2 MAS 2 ES 5
2 MAS 2 ES 5
2 MAS 2 ES 5
2 MAS 2 ES 5
2 MAS 2 ES 5
2 MAS 2 ES 5
2 MAS 2 ES 5
¡SOLO BROMEO!

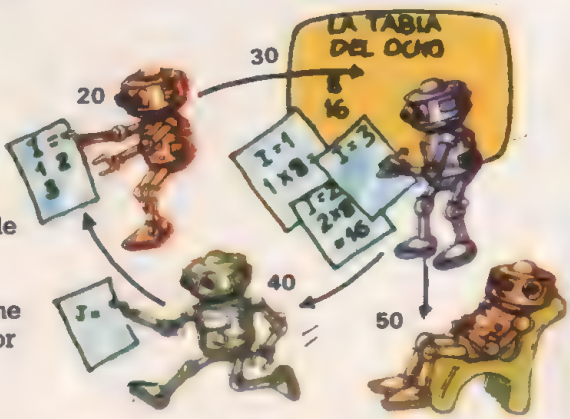
En este programa, el bucle de los pasos 10 a 30 hace que la computadora repita ocho veces el paso 20. Cada vez que pasa por el paso 20 imprime la misma suma tonta.

Después de hacerlo ocho veces la computadora continúa con el resto del programa. El paso 40 sólo deja una línea en blanco.

3 Programa de la tabla del ocho

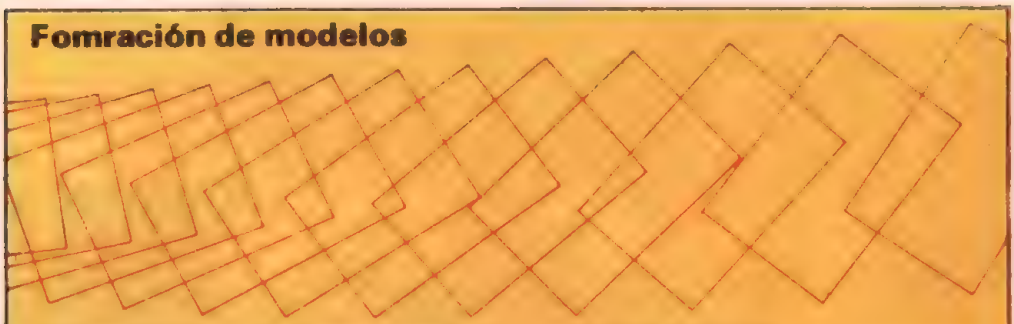
```
10 PRINT "LA TABLA DEL OCHO"
20 FOR I=1 TO 12
30 PRINT I*8
40 NEXT I
50 END
```

Bucle



Aquí se utiliza I para contar el número de bucles y también cómo parte de la operación I*8. El paso 20 inicia I a 1, luego a 2, 3, etc., hasta 12. El paso 30 tiene el valor actualizado de I, lo multiplica por 8 y escribe la respuesta. Entonces el paso 40 manda a la computadora que vuelva al paso 20 para buscar el siguiente valor de I.

Fomración de modelos



Los bucles *FOR ... NEXT* son útiles para hacer modelos, como éste, de una configuración simple repetida muchas veces. El programa para este modelo es demasiado largo para escribirlo aquí en su totalidad, pero sería parecido a esto:

```
10 FOR I=1 TO 45
20 DIBUJA UN RECTANGULO Y
   VARIA SU POSICION
   LIGERAMENTE CADA VEZ.
30 NEXT I
40 END
```

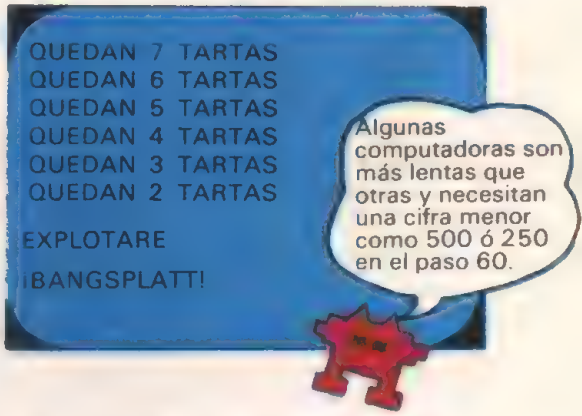
Pasos

Algunas veces es útil incrementar el valor de I en pequeñas cantidades que no sean uno. Por ejemplo, puedes desear subir de tres en tres o bajar de siete en siete. Para hacer esto utilizarás la palabra *STEP*. En el siguiente programa *STEP-1* hace bajar I uno cada vez que la computadora pasa a través del bucle entre los pasos 10 y 40.

Programa de la computadora glotona

El número 2 para el bucle después de I=2. Esto es cuando queda una sola tarta.

```
5 CLS
10 FOR I=7 TO 2 STEP-1
20 PRINT "HAY"; I; "TARTAS SIN COMER"
30 NEXT I                    Bucle
40 PRINT
50 PRINT "EXPLOTARE"
60 FOR K=1 TO 1000
70 REM: NO HAGAS NADA
80 NEXT K                    Bucle
90 PRINT
100 PRINT "BANGSPLATT"
```



Hay dos bucles es este programa. El que se encuentra entre los pasos 20 y 30 hace que la computadora imprima el paso 20 seis veces. Cada vez, el valor de I es reducido en uno y la cifra de I se imprime en el paso 20. En el bucle que va del paso 60 al 80 la computadora no tiene que hacer nada. Sólo recorre todos los valores de K entre 1 y 1000 y esto hace detenerse un momento. Los pasos que empiezan con REM (abreviatura de REMARK) son ignorados por la computadora y son útiles para recordar lo que el programa está haciendo.

Ejercicios

1. ¿Puedes alterar el programa de la tabla del ocho de la izquierda para hacer el saque por pantalla "1×8=" a la vez que la respuesta?
2. ¿Puedes escribir un programa para la tabla del "N" que sea un programa que calcule la tabla para cualquier número que suministres a la computadora? Primero necesitas que la computadora te pregunte por un número N, luego utiliza un bucle para calcularlo y saca en pantalla las tablas. Si quieres incluye algunos pasos al final del programa para que te pregunte si quieres las tablas para otro número, y el programa se repite el solo.

Trucos con bucles

Aquí hay algunos programas más que usan bucles. Abajo puedes encontrar cómo usar bucles, dentro de bucles, para repetir algunas cosas al mismo tiempo. Estos son llamados bucles anidados.

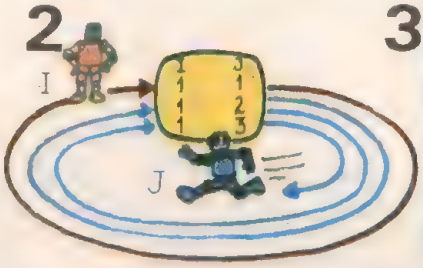
1 Bucles anidados

```
5 PRINT "I"; "J"
10 FOR I=1 TO 3
20 FOR I=1 TO 3
30 PRINT I,J
40 NEXT I
50 NEXT I
60 END
```

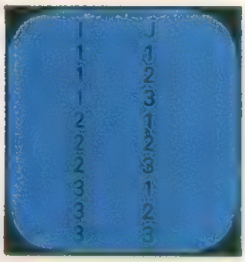
Bucle J

Bucle I

2



3



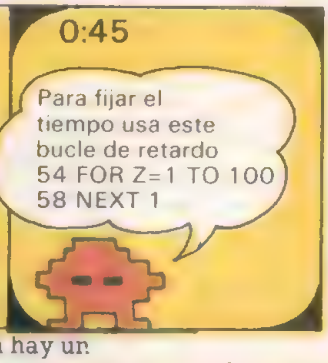
Este programa tiene un bucle I y un bucle J. El bucle I está anidado dentro del I, y por cada vez que se realiza el bucle I, el bucle J se repite tres veces, imprimiendo el valor de J cada vez. El dibujo de abajo muestra el resultado de este programa; las comas espacian las figuras de la manera indicada.

Reloj computador

```
5 CLS
10 LET M=0
20 LET S=0
30 FOR M=0 TO 59
40 FOR S=0 TO 59
50 PRINT M; ":"; S
60 CLS
70 NEXT S
80 NEXT M
90 END
```

Bucle de segundos

Bucle de minutos



Dentro de una computadora hay un "reloj" electrónico que fija el ritmo para todo el trabajo de la computadora. Este reloj origina entre una y cuatro millones de pulsaciones por segundo. Este programa hace a la computadora como un reloj digital. Tiene bucles anidados: uno para contar los segundos y otro para contar los minutos. El bucle de los segundos es recorrido 59 veces por cada vez que se recorre el bucle de los minutos. Si intentas realizar este programa en una computadora, éste puede ir muy deprisa en un principio. Necesitarás poner un "bucle de retardo" extra. Luego fijarlo cambiando la cifra en el bucle para que tu reloj computador "funcione" a la misma velocidad que uno real.

Errores en bucles

```
10 FOR I=1 TO 4
20 FOR J=1 TO 4
30 PRINT I
40 PRINT J
50 NEXT I
60 NEXT J
```

Las dos partes de un bucle anidado deben estar una dentro de la otra.

Comprobador de números aleatorios

```
10 FOR I=1 TO 1000
20 FOR R= INT (RND(1)*6+1)
30 IF R=1 THEN LET A=A+1
40 IF R=2 THEN LET B=B+1
50 IF R=3 THEN LET C=C+1
60 IF R=4 THEN LET D=D+1
70 IF R=5 THEN LET E=E+1
80 IF R=6 THEN LET F=F+1
90 NEXT I
100 PRINT "TERMINADO"
110 PRINT A, B, C
120 PRINT D, E, F
130 END
```

Este programa tarda bastante tiempo; puedes acortarlo cambiando el número del paso 10 a 500 o incluso a 250.

```
RUN
TERMINADO
162    168    167
160    187    156
```

Este programa comprueba si RND realmente funciona. El bucle del paso 10 al 90 hace que la computadora escoja un número aleatorio entre 1 y 6 mil veces. Lleva la cuenta del número de veces que cada número es elegido. *

* En algunas computadoras, por ejemplo, la ZX81 y la micro BBC, necesitas algunos pasos extras al principio del programa para hacer que cada variable sea igual a cero.

Programa de repetición de un modelo

Este programa utiliza bucles anudados para repetir un pequeño modelo en toda la pantalla. El programa parece muy complicado, pero si lo lees detenidamente y comprendes lo que hace comprenderás pronto como funciona. La figura del dibujo se decide mediante números aleatorios y será diferente cada vez que ejecutes el programa.

Para computadoras con una elevada resolución gráfica utiliza números aleatorios grandes. Ej.: en BBC cambia el índice de las líneas 10 a 40 por 60.

```
5  CLS
10 LET A=INT (RND(1)*6+1)
20 LET B=INT (RND(1)*7+1)
30 LET C=INT (RND(1)*6+1)
40 LET D=INT (RND(1)*4+1)
50 INPUT "CUANTOS PUNTOS
   HORIZONTALES"; W
60 INPUT "CUANTOS VERTICALES"; V
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6
90 PLOT (J+A, I+B)
100 PLOT (J+A, I+C)
110 PLOT (J+C, I+D)
120 PLOT (J+B, I+D)
130 NEXT J
140 NEXT I
150 END
```

Bucle J

Bucle I

Estos pasos eligen números aleatorios para el dibujo y los almacena en A, B, C y D.

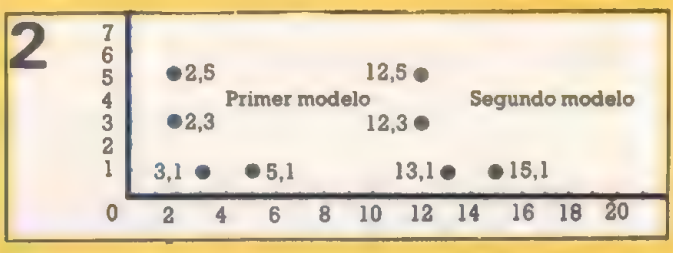
Los pasos 50 y 60 preguntan por la anchura (W) y la altura (V) de su pantalla.

El bucle I cuenta el número de veces que se repite el modelo de la pantalla. Cada vez I se incrementa según la altura de la pantalla (V) dividida por 6, de manera que el modelo se repite 6 veces en la pantalla.

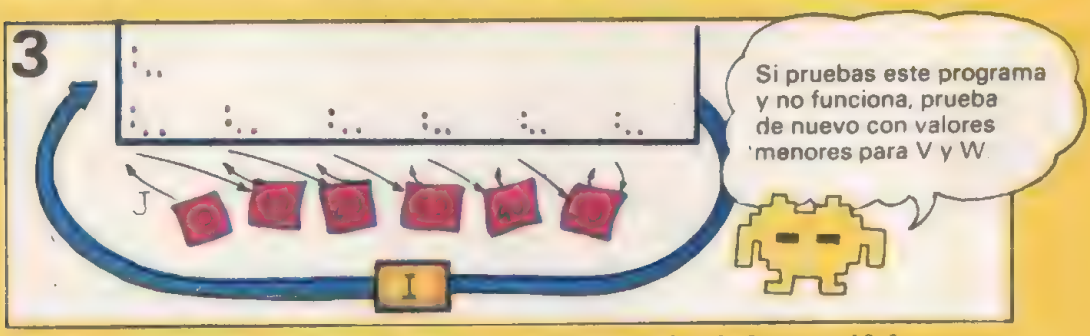
Cada vez que se repiten los bucles, los pasos 90 a 120 le dicen a la computadora que encienda cuatro pixels utilizando los valores actuales de I y J más los números aleatorios.

El bucle J cuenta el número de veces que se repite el modelo a través de la pantalla. Funciona igual que el bucle I.

Imagina que la computadora ha elegido los números aleatorios 2, 5, 3 y 1 y que el ancho y la altura de la pantalla son ambos 60.



En el primer recorrido del programa I y J son 0, así que la computadora realiza el primer modelo utilizando sólo los números aleatorios. La línea 130 le manda a buscar el siguiente valor de J que es J+60/6, esto es 10. Entonces dibuja el segundo modelo a lo largo de la pantalla.



La computadora repite el bucle J seis veces, añadiendo cada vez 10 a J y dibujando así el modelo a lo largo de la pantalla. Vuelve entonces a buscar el siguiente valor de I que es 10. J se pone a 0 de nuevo y la computadora dibuja el siguiente paso utilizando 10 para I e incrementando J por 10 cada vez.

Ejercicio.—¿Podrías escribir un programa para repetir modelos que repita la figura de un invasor espacial en la pantalla? Hay algunas pequeñas sugerencias de ayuda en la página 45.

Subrutinas

Una subrutina es una especie de mini-programa dentro del programa. Lleva a cabo una tarea particular, como sumar números o llevar una puntuación. Y puedes mandar a la computadora a ella cuando quieras que se realice esta tarea. Esto te libra de escribir los pasos del programador cada vez y hace que el programa sea más corto y fácil de leer e introducir en la computadora.

Imagina que tienes un robot que te ayuda, él cual puedes programar para que te haga recados. Si quisieses algo de la tienda, tendrías que darle instrucciones precisas diciéndole como llegar hasta allí.

Cada vez que quisieras que el robot te comprase algo le tendrías que dar las mismas instrucciones. Sería mucho más simple darle al robot una subrutina de compras y decirle que se remitiese a ella cada vez.

4 Programa de compras

```
10 PRINT "QUE DESEA DE LA TIENDA"
20 INPUT X$
30 GOSUB 100
40 PRINT "NADA MAS"
50 INPUT M$
60 IF M$="SI" THEN GOTO 10
70 STOP

100 REM: SUBRUTINA DE COMPRAS
110 PRINT "SAL, TUERCE A LA IZQUIERDA"
120 PRINT "DE NUEVO A LA IZQUIERDA,
    ENTRA EN LA TIENDA"
130 PRINT "COMPRA"; X$; "VEN A CASA"
140 RETURN
```

El paso 30 manda al computador al primer paso de la subrutina.

Necesitas la palabra STOP al final del programa principal para evitar que la computadora entre de nuevo en la subrutina.

Es útil etiquetar una subrutina con un paso *REM* para que sepas para qué sirve.

Esta manda a la computadora al paso 40 – El paso siguiente a *GOSUB*.

Si olvidas el paso de *RETURN* obtendrás un error.

En *BASIC* para decirle a la computadora que vaya a una subrutina utiliza la palabra *GOSUB* con la palabra *RETURN* al final de la subrutina. *GOSUB* deberá ir seguida de el número del paso de la subritina. *RETURN* no necesita un número de paso. La computadora automáticamente regresa a la instrucción en la que dejó el programa principal. Puedes mandar a la computadora a una subrutina cualquiera del programa, cuando quieras.

Programas GOSUB

Una subrutina es útil para llevar a cabo una tarea que se quiere repetir varias veces en diferentes partes del programa. Aquí hay algunos programas con subrutinas.

Programa de números

```
 50 INPUT A
 60 INPUT B
 70 GOSUB 250
 80 PRINT "A DIVIDIDO POR B="; A/B
 90 GOTO 50
250 REM: SUBRUTINA PARA ACABAR
260 IF A=0 AND B=0 THEN STOP
270 RETURN
```

Esta subrutina proporciona una salida al programa. Si deseas pasar, debes introducir 0 en los pasos 50 y 60. Este programa no necesita un STOP antes de la subrutina porque el paso 90 le hace regresar.

Programa de conversión

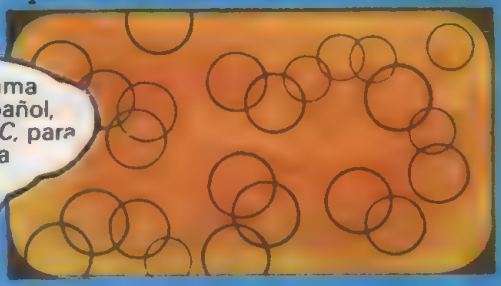
```
100 INPUT "DISTANCE"; M
110 INPUT "TIEMPO"; T
120 GOSUB 200
130 PRINT "VELOCIDAD MEDIA ES"
140 PRINT M/T; "MPH Y"; K/T; "KPH"
150 STOP
200 REM: SUBRUTINA PARA CONVERTIR MILLAS
210 LET K=M*1,609
220 RETURN
```

Esta es una subrutina para convertir millas en kilómetros. A menudo se puede usar la misma subrutina en diferentes programas. Comprueba, sin embargo, que utiliza el mismo nombre de variable.

Programa de círculos

```
 1 Centro del círculo = X,Y
 2 Radio del círculo = R
 3 Color = X
 4 Gosub 10
 5 Ir a 1
10 REM: Subrutina para dibujar círculos
11 Dibujar un círculo con centro
   X,Y; radio R y color X.
12 Volver
```

Este programa está en español, no en *BASIC*, para dar una idea general.



La subrutinas son útiles en programas de gráficos como éste de dibujar diagramas con números creados en la parte principal del programa. Con este programa puedes dibujar muchos círculos diferentes dándole a la ordenadora información diferente en las líneas 1 a 5.

Programa *QUIZ*

```
 5 LET C=0
10 PRINT "CUANDO FUERON ESTAS
   COSAS INVENTADAS"
20 READ C$, F
30 PRINT C$
40 INPUT A
50 LET C=C+1
60 IF C=3 THEN STOP
70 GOSUB 100
80 GOTO 10
100 REM: SUBRUTINA DE RESPUESTAS
110 IF ABS (A–F) < 10 THEN PRINT "OK"
120 IF ABS (A–F) > 10 THEN PRINT "OK"
130 PRINT "PRUEBE OTRO"
140 RETURN
200 DATA TELEFONO, 1876, IMPRESORA, 1450, BICICLETA, 1791
```

En el paso 20 la computadora busca la línea de datos e introduce la primera palabra en C$ y el primer número en F.

Este paso escribe la palabra que hay en C$.

El contador C detiene el programa después que se ha repetido tres veces, ya que sólo hay tres datos para C$ y F.

Esta es la subrutina.

Cada vez que se repite el programa, las palabras y los números de C$ y F son reemplazados por el siguiente par de datos.

Este programa utiliza una subrutina para comprobar las respuestas. Las respuestas correctas se almacenan en F y las respuestas de la persona en A. En los pasos 100 a 110 de la subrutina, la computadora compara A y F. La palabra ABS significa "absoluto", y hace que la computadora compruebe la diferencia entre los valores de A y F (ignora el signo menos). Si la diferencia es menor que 10, escribirá "OK". Si es mayor de 10, escribirá "NO".

Haciendo cosas con palabras

La mayoría de las computadoras pueden examinar las palabras almacenadas por variables y hacer varias cosas con ellas. Pueden comprobar el contenido de una variable y ver si contiene una palabra o letra particular. Esto es útil para comprobar las palabras metidas por alguien que usa el programa. Las computadoras también pueden reorganizar las letras o palabras en un orden distinto y añadirlas a letras de otras variables. Abajo puedes ver cómo se hacen esas cosas en *BASIC*.

1

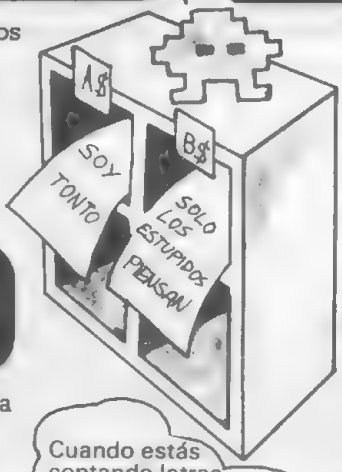
```
10 A$= "YO SOY TONTO"
20 B$= "SOLO LOS TONTOS"
30 C$=B$+A$
RUN
SOLOS LOS ESTUPIDOS PIENSAN QUE
YO SOY TONTO
```

En la mayoría de las computadoras, pero no en el ZX81 puedes omitir la palabra *LET*.

2

```
PRINT LEFT$(B$,4)
SOLO
PRINT LEFT$(B$,4)+ +A$
SOLO YO SOY TONTO
```

Puedes sumar los contenidos de dos variables así: necesitas el espacio entre signos de puntuación para dejar un espacio entre las palabras.



También puedes añadir parte de las variables de esta forma. LEFT$(B$,4) quiere decir que tomes las cuatro letras primeras de la izquierda de BS.

3

```
PRINT RIGHT$(A$,8)
TONTO
```

Para decir a la computadora que utilice letras de la derecha debes usar RIGHT$ con el nombre de la cadena y el número de letras que quieres.

4

```
PRINT MID$(B$,10,6)
TONTOS
```

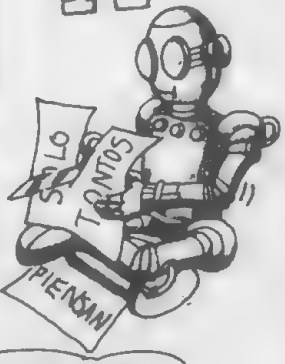
Este dice a la computadora que tome las palabras de en medio. El primer número indica dónde debe empezar y el segundo cuantas letras tomar.

Cuando estás contando letras, cuenta espacios y puntuación también.

5

```
10 K$= "DING DONG"
20 PRINT LEN(K$)
RUN
 9
```

También puedes hallar el tamaño de una cadena – el número de letras, espacios y símbolos que contiene. Para hacer esto debes usar *LEN*.



IF A$="COMPUTER BOOK"
¿qué es LEFT$(A$,8)?
¿RIGHT$(A$,10)?
¿MID$(A$,5,8)?

Nota para usuarios del Sinclair

PRINT A$ (8 TO 14)
ESTUPIDO
PRINT B$(17 TO 19)
PIE

Esto quiere decir que tomas las letras desde la número 6 a la 11.

Las computadoras *SINCLAIR* no utilizan *LEFT$*, *RIGHT$* y *MID$*, pero puedes decirle a la computadora que tome las palabras que desees en la forma indicada arriba.

Programa codificador

Este programa codifica las palabras automáticamente. Los servicios de inteligencia utilizan programas similares, pero más complejos, para construir códigos.

La manera más fácil de entender este programa es escribir un mensaje secreto en un papel y recorrer los pasos del programa llevando a cabo las tareas de la computadora en tu propio mensaje y escribiéndolas.

```
5  LET C$=" "
7  LET D$=" "
10 PRINT "TECLEAR UN MENSAJE BREVE"
20 INPUT M$
30 PRINT "AHORA TECLEAR UN NUMERO
   SECRETO ENTRE 2 Y"; LENS(M$)-1
40 INPUT N
50 LET A$=RIGHT$(M$,N)
60 LET B$=LEFT$(M$,LEN(M$)-N)
70 LET M$=A$+B$
80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$=CS+MID$(M$,I,1)
100 NEXT I
110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$,J,1)
130 NEXT J
140 LET M$=C$+D$
150 PRINT "EL MENSAJE CODIFICADO ES"
160 PRINT M$
170 END
```

Deja una variable vacía de cadena.

Esto es la longitud de tu mensaje menos 1.

N (tu número secreto) letras desde la derecha M$.

La longitud de M$ menos N número de letras desde la izquierda de M$ (esto es el resto de las letras).

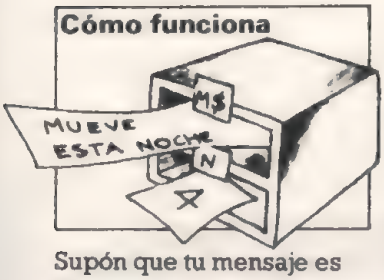
Reemplaza las letras de M$ con las de A$+B$.

Desde 1 hasta el número de letras de tu mensaje de dos en dos; esto es, 1, 3, 5, etc. Cada vez que el bucle I se repite el paso 90 toma una letra de la posición I de M$ y la pone en C$.

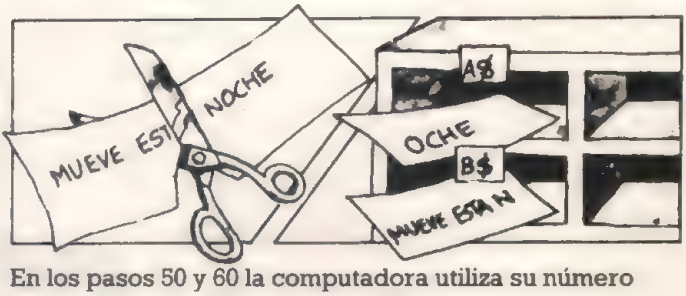
Desde 2 hasta el número de letras de tu mensaje de dos en dos; esto es, 2, 4, 6, etc. Funciona igual que el bucle I.

Reemplaza las letras de M$ otra vez.

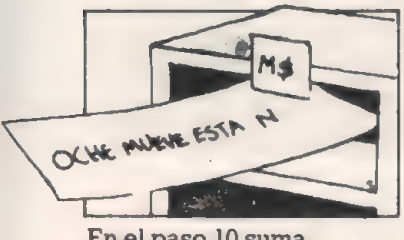
Cómo funciona



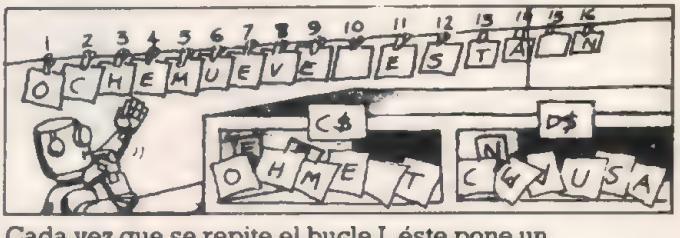
Supón que tu mensaje es "mueve esta noche" y su número secreto 4. Estos se almacenan en M$ y N.



En los pasos 50 y 60 la computadora utiliza su número secreto para dividir el mensaje. En el paso 50 toma cuatro letras de la derecha del mensaje y las pone en A$. En el paso 60 pone el resto en B$.



En el paso 10 suma A$ y B$. Esto pone las letras finales del mensaje al principio.



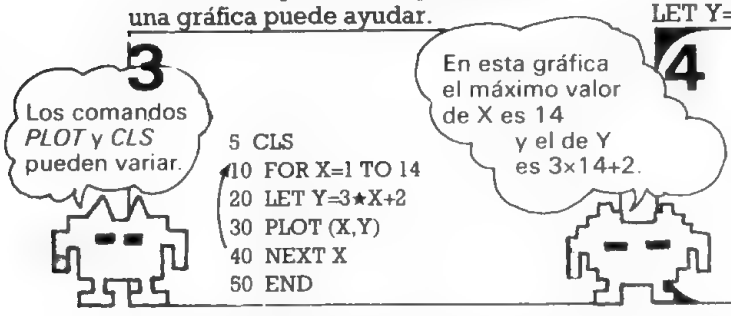
Cada vez que se repite el bucle I, éste pone un número-impar de letra en C$ (O, H, M...). Cada vez que el bucle J se repite este pone un número par de letra en D$ (C, E, U...). Entonces suma C$ y D$ para construir el mensaje codificado.

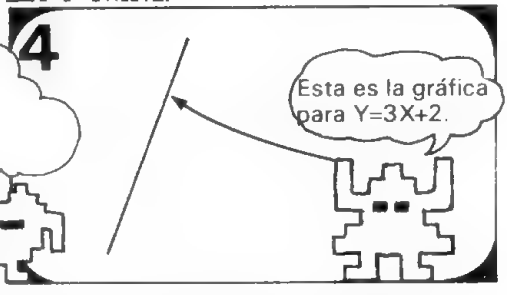
Gráficas y símbolos

Se puede programar una computadora para que presente información en muchos tipos de formas distintas; por ejemplo, palabras, números, dibujos o gráficas. La información muy complicada puede simplificarse si se ilustra con gráficos, dibujos y símbolos.

Imagina un melocotonero cuya producción de frutas incrementa cada año en relación con su edad. Esto puede expresarse como una ecuación Y=3X+2 (Y es la entrega, X la edad). Sin embargo, es difícil comprender lo que significa y una gráfica puede ayudar.

Con una computadora es fácil dibujar una gráfica de la forma en que Y varía en relación a X. Para dibujar la gráfica necesitas hallar el valor de Y para cada valor de X. Puedes hacer esto fácilmente en un programa utilizando la sentencia LET Y=3★X+2.





Este es el programa para dibujar la gráfica. El bucle hace a X pasar por todos los valores de 1 hasta 14. Cada vez que el bucle se repite el paso 20 utiliza el valor de X. Para calcular Y y el paso 30 dibuja X e Y en la pantalla. En programas de gráficos debes asegurarte de que los valores máximos de X e Y encajarán en la pantalla u obtendrás un error.

Computadoras y matemáticas

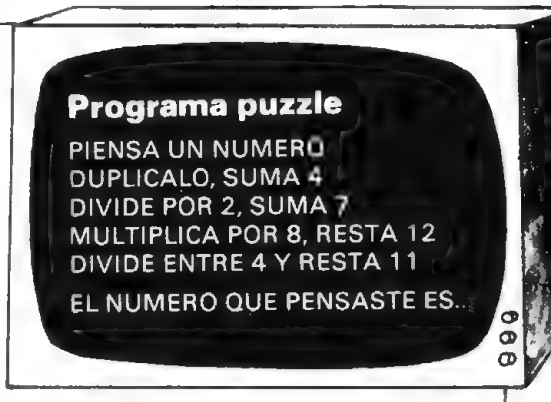
En cálculos que tienen varias partes, como 3×X+2, la computadora siempre multiplica y divide antes de sumar y restar. Esto quiere decir que la computadora daría la misma respuesta a estas dos sumas:

```
PRINT 4★6+8          PRINT 8+4★6
32                   32
```

Si deseas que la computadora realice las operaciones en un orden diferente debes usar paréntesis así:

```
PRINT (8★4)★6
72
```

Esta vez la computadora suma 8 y 4 y luego multiplica por 6.

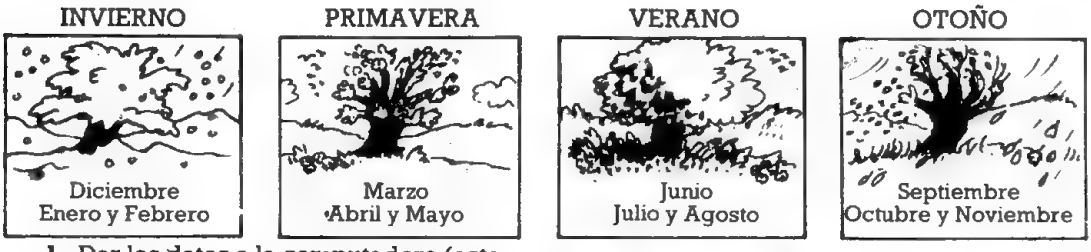


¿Puedes escribir un programa para que la computadora lleve a cabo este conocido truco? (Para encontrar el número debes restar 4 del resultado y después dividir por 2.)

Programa de cumpleaños

Este programa utiliza otra forma de mostrar la información en la pantalla. Utiliza símbolos para comparar el número de personas que han nacido en las diferentes estaciones del año. Se podría usar un programa así para comparar las veces que se ve un determinado pájaro en las diferentes estaciones o el número de veces que ganan diferentes equipos de fútbol. Antes de escribir un programa tan largo como éste sería buena idea trazarse un plan.

Objetivo: Comparar el número de personas con cumpleaños en invierno, primavera, verano y otoño.



1. Dar los datos a la computadora (esto es, las estaciones en las que ha nacido la gente) para un total de 20 personas.
2. Almacenar los datos en la computadora.
3. Sacar los datos por pantalla.


```
5 LET A=0
6 LET B=0
7 LET C=0
8 LET D=0
10 FOR I=1 TO 20
20 PRINT "PERSONA"; I; "NACIDO EN"
30 PRINT "INVIERNO, PRIMAVERA, VERANO, OTOÑO"
40 PRINT "TIPO IN, PR, VE, OT"
50 INPUT B$
60 IF B$="IN" THEN LET A=A+1
70 IF B$="PR" THEN LET B=B+1
80 IF B$="VE" THEN LET C=C+1
90 IF B$="OT" THEN LET D=D+1
100 NEXT I
110 PRINT "TOTAL INVIERNO"
115 LET N=A
120 GOSUB 200
130 PRINT "TOTAL PRIMAVERA"
135 LET N=B
140 GOSUB 200
150 PRINT "TOTAL VERANO"
155 LET N=C
160 GOSUB 200
170 PRINT "TOTAL OTOÑO"
175 LET N=D
180 GOSUB 200
190 STOP
200 REM: SUBRUTINA IMPRIMIR ESTRELLAS
210 IF N=0 THEN GOTO 250
220 FOR I=1 TO N
230 PRINT "★"
240 NEXT I
250 PRINT
260 RETURN
```

Variables vacías preparadas para almacenar los totales de cada estación.

Bucle para hacer que la computadora pregunte una vez por cada persona del total.

Pasos 60 a 100 para comprobar la respuesta que está en B$ y añadir un uno a la variable de la estación correspondiente.

Manda a la computadora a preguntar de nuevo.

La subrutina imprime un número de estrellas igual al número de cada variable.

Poniendo el total en N cada vez, la computadora puede utilizar la misma subrutina para cada estación.

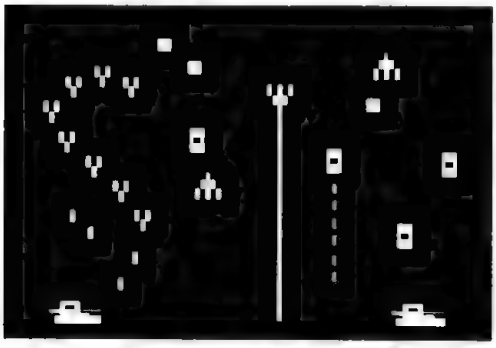
Obliga a la computadora a mantenerse en la misma línea para imprimir las estrellas.

El paso 210 comprueba el caso que en alguna estación no haya nacido nadie.

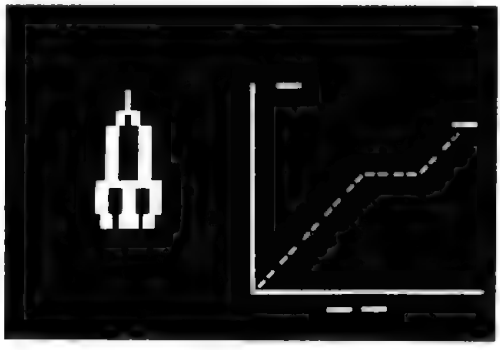
El programa principal carga en N el total de A, B, C o D. El bucle obliga a la computadora a ejecutar el paso 230 "N" veces.

Más gráficas

Estas dos páginas muestran cómo puedes utilizar *PLOT* y *UNPLOT* para hacer dibujos móviles en la pantalla. Los dibujos móviles se llaman gráficos vivos y son útiles para programas de juegos o para ilustrar programas que explican, por ejemplo, los principios gravitatorios o balísticos y recorridos de huida.



Las imágenes de video y los juegos electrónicos se controlan con un pequeño computador. La computadora se programa para jugar, sólo los juegos y los programas están en el código.



Una microcomputadora de propósitos generales programada en *BASIC* hace dibujos más simples y más despacio. No puede manejar todas las instrucciones de pantalla con la suficiente rapidez.

Programa *PLOT/UNPLOT*

1

```
10 LET X=1
20 LET Y=1
30 PLOT (X,Y)
40 UNPLOT (X,Y)
50 LET X=X+1
60 LET Y=Y+1
70 GOTO 30
```

Este programa breve produce un movimiento de luz a través de la pantalla. Recuerda, los comandos de *PLOT* y *UNPLOT* varían según las diferentes computadoras.

Cuando el punto alcanza el borde de la pantalla, el programa puede parar lanzando un mensaje de error, ya que los valores de X e Y están fuera de pantalla

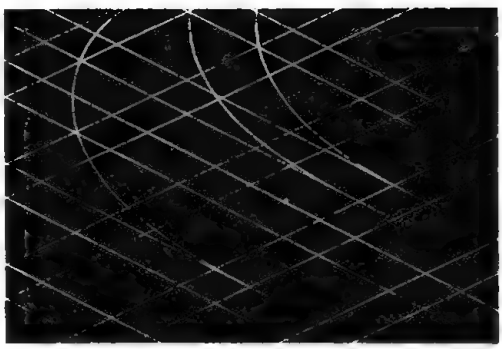
3

Los juegos de video de bate y pelota utilizan programas como el de arriba para mover la pelota en la pantalla. Hay instrucciones de programas simples para mantener la pelota moviéndose cuando alcanza el borde de la pantalla.

Cuando la pelota alcanza el borde superior de la pantalla, la cantidad que es sumada habitualmente a Y en este caso se resta. De la misma manera cuando alcanza el borde derecho, la cantidad se resta a X.

Programa de modelos lineales

Este programa traza una línea a través de la pantalla y cuando alcanza los bordes la devuelve en otra dirección. No utiliza *UNPLOT* de modo que las líneas dejan un dibujo en la pantalla. El dibujo de la derecha muestra lo que ocurre cuando corremos este programa. El programa está resuelto para encender 10.000 pixels en el paso 100. Puedes variar esta figura para hacerla menor o cambiar el programa para otra figura que prefiera.



```
10 REM: TRACE MODELOS GRAFICOS AQUI SI ES NECESARIO
20 PRINT "¿CUANTOS PIXELS HORIZONTALES?";
30 INPUT H
40 PRINT "¿Y VERTICALES?"
50 INPUT V
55 CLS
60 LET X=H/2
70 LET Y=V/2
80 LET S=1
90 LET T=1
100 FOR I=1 TO 10000
110 LET S=S+(INT(RND(1)*10+1)-5)/50
120 LET X=X+S
130 LET Y=Y+T
140 IF X<S THEN LET S=-S
150 IF X>H-S THEN LET S=-S
160 IF Y<S THEN LET T=-T
170 IF Y>V-S THEN LET T=-T
180 GOSUB 300
190 NEXT I
200 STOP
300 REM: PLOT LINEA
310 PLOT (X,Y)
320 RETURN
```

Pasos 20 a 50: pregunta por la altura y el ancho de la pantalla. El punto y coma coloca su respuesta en la misma línea que la pregunta.

Esto obliga a X e Y a comenzar en el centro de la pantalla.

S y T son las cantidades que se añadirán a X e Y para que la línea se mueva.

El bucle de pasos 100 a 190 se repite 10.000 veces. Cada vez X e Y son variadas en una pequeña cantidad.

Esto proporciona un número muy pequeño para añadir a X. El número varía cada vez que se repite el bucle.

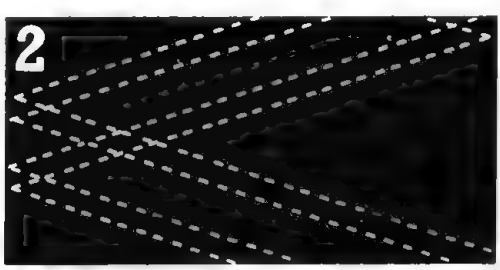
Estos pasos comprueban los extremos y cambian S y T cuando X e Y llegan a estar a cinco pixels de distancia del borde.

Manda a la computadora a la subrutina para dibujar la línea.

Ilumina los pixels con el valor actual de X e Y.

Experimentos

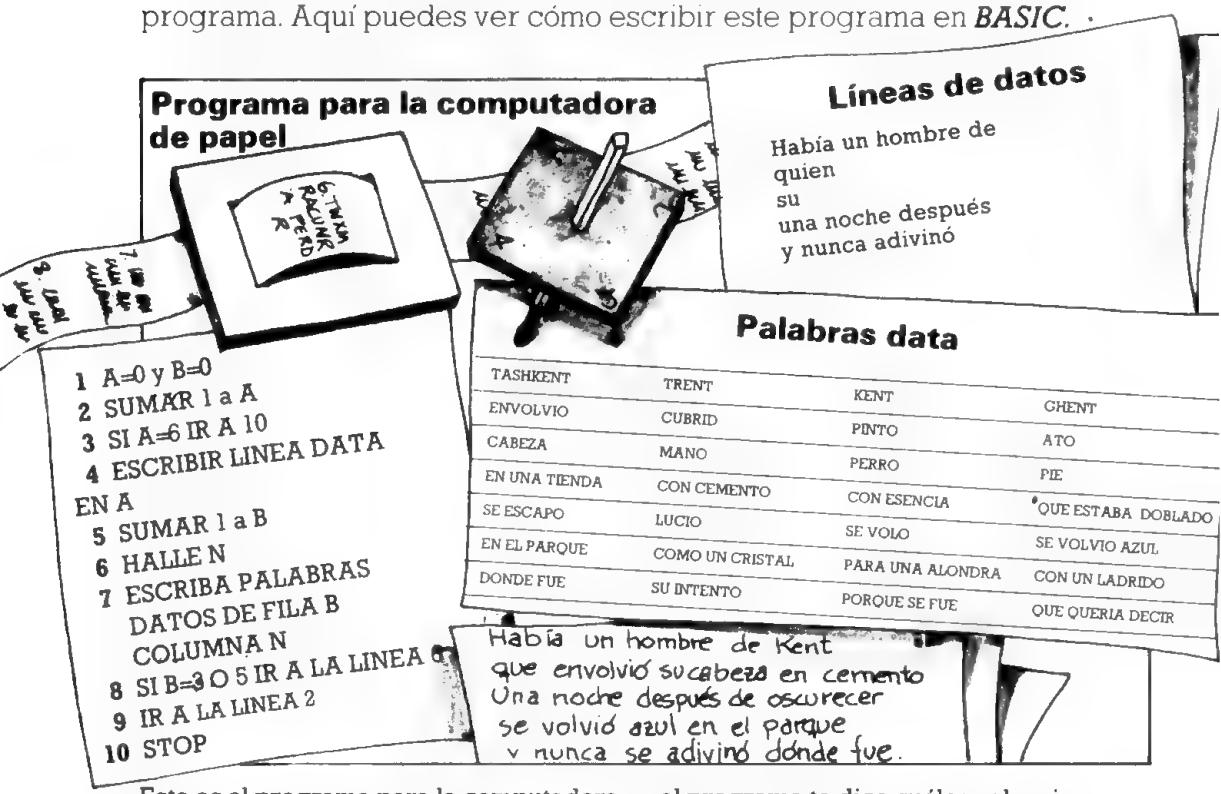
El paso 110 suma una pequeña cantidad aleatoria a X cada vez y esto hace que la línea se deslice a través de la pantalla. Si tienes una computadora, trata de borrar este paso. Las líneas en la pantalla deberían volverse paralelas.



Intenta variar los números de los pasos 80 y 90 por, digamos 5 ó 10 (o mayores en una computadora con alta resolución de gráficos). Esto obliga a la computadora a encender pixels a intervalos.

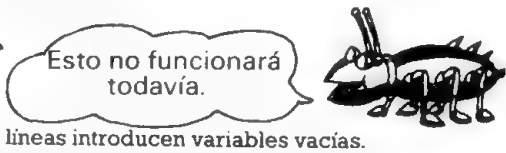
Programas de cuentos

Las próximas páginas te enseñarán cómo constituir un programa que puede componer muchos cuentos. Este libro enseña cómo hacer una computadora de papel que utiliza una versión más simple de este programa. Aquí puedes ver cómo escribir este programa en *BASIC*.



Este es el programa para la computadora de papel. Parece *BASIC*, pero no funcionaría en una verdadera computadora. Las palabras para el cuento se almacenan en tiras de papel y el programa te dice cuáles seleccionar. El número giratorio es un número aleatorio generador de números aleatorios entre 1 y 4.

1 Traducción del programa a *BASIC*




```
10 LET A=0
20 LET B=0
30 LET A=A+1
40 IF A=6 THEN STOP
50 ESCRIBE LINEA DATA A
60 LET B=B+1
70 LET N=INT(RND(1)*4+1)
80 ESCRIBE PALABRAS DATA
   DE FILA B COLUMNA N
90 IF B=3 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

Estas líneas introducen variables vacías.

Los pasos 30 y 40 llevan la cuenta del número de datos que ha seleccionado la computadora.

Los pasos 50 y 80 no están en *BASIC* todavía.

El paso 60 lleva cuenta del número de datos de palabra.

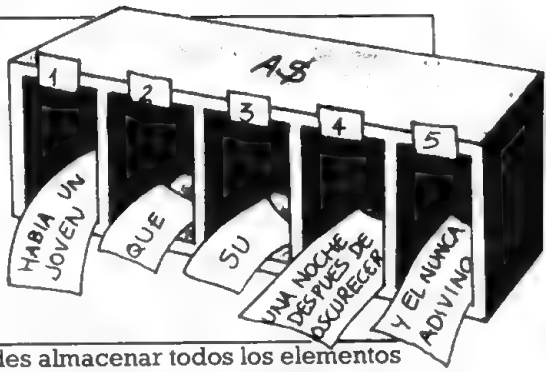
Crea un número aleatorio entre 1 y 4.

Los pasos 90 y 100 le mandan a seleccionar otra línea de datos.

La mayor parte del programa es fácil de traducir a *BASIC*, pero las líneas 50 y 80 son más fáciles. El ordenador precisa una forma de almacenar y tomar las líneas de datos y palabras que hacen falta para cada línea del cuento.

2 Dando datos a la computadora

```
50 READ A$
.
.
180 DATA HABIA UN JOVEN DE; QUIEN; SU
190 DATA UNA NOCHE DESPUES DE
    OSCURECER,
    Y EL NUNCA ADIVINO
```



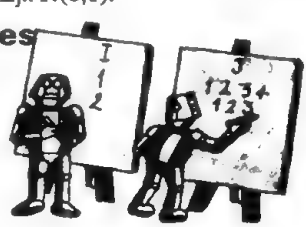
Para dar a la computadora las líneas de datos y palabras puedes utilizar *READ ... DATA*. Cada vez que la computadora lleva a cabo la instrucción de leer toma otro elemento de *DATA* y lo almacena en la variable.

Puedes almacenar todos los elementos de datos en una gran variable llamada A$. Una variable que contiene más de un elemento se llama matriz, y a cada elemento se hace referencia por un número.

3

Una variable puede contener varias filas de datos y se pueden almacenar todos los datos de palabras en una variable así. Se llama matriz de dos dimensiones. Aquí a cada dato se hace referencia por el número de fila y columna en que se encuentra. Así, READ B$ (4,2) da CON CEMENTO y READ B$ (6,3) da PARA UNA ALONDRA. Puedes almacenar números en las matrices también utilizando una variable numérica. Ej.: N(5,7).

4 Introducir data en las variables




```
10 FOR I=1 TO 7 ]—— I es el número de fila
20 FOR J=1 TO 4 ]—— J es el número de columna
30 READ B$ (I,J)
40 NEXT J
50 NEXT I
60 DATA TASHKENT, TRENT, KENT, GHENT
70 DATA ENVOLVIO, CUBRID, PINTO, ATO
80 DATA CABEZA, MANO, PERRO, PIE
90 DATA EN UNA TIENDA, CON CEMENTO, CON ESENCIA, QUE ESTABA DOBLADO
100 DATA SE ESCAPO, LUCIO, SE VOLO, SE VOLVIO AZUL
110 DATA EN EL PARQUE, COMO UN CRISTAL, PARA UNA ALONDRA, CON UN LADRIDO
120 DATA DONDE FUE, SU INTENTO, POR QUE SE FUE, QUE QUERIA DECIR
```

Para introducir cada dato en la variable necesitas ser capaz de alterar los números entre paréntesis del *READ*. Puedes hacer esto con bucles. B$ necesita bucles anidados como se indica arriba con un bucle I para el número de fila y un bucle J para el número de columna. Cada vez que el bucle I se ejecuta, el bucle J se ejecuta cuatro veces, una para cada columna de la fila.

* Las computadoras *SINCLAIR* tienen un tratamiento distinto de las variables, y este programa no funcionaría en un *SINCLAIR*. Hallarás sobre esto al volver la página.

5 Creando espacio a las variables

```
5  DIM K$(5)
10 FOR I=1 TO 5
20 READ K$(I)
30 NEXT I,
40 STOP

60 DATA PERRO, GATO, SAPO, CHINCHE
```

Este es el tamaño de la variable. Esto es 5 elementos en una fila.

Esta línea pone el dato en K$ cada vez que se repite el bucle.

Al principio del programa debes decir a la computadora lo grande que quieres sea la variable. Esto se hace con la palabra *DIM* seguida del nombre de la variable y el número de elementos del dato, ej.: DIM K$(5).

Para una matriz bidimensional debes dar a la computadora el número de filas y columnas de la variable. Por ejemplo, DIM C$(5,3). Debes tener siempre el número exacto de datos para la variable u obtendrá un error.

6 Imprimiendo datos

```
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$ (A)
250 LET B=B+1
260 LET N=INT(RND(1)★4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

A lleva la cuenta del número de veces que está sección del programa se repite.

B lleva la cuenta de las filas de datos y se asegura de que la fila correcta se utiliza en cada línea de datos.

Los pasos 280 y 290 hacen que la computadora imprima las palabras de otra línea de datos antes de imprimir la siguiente línea de datos.

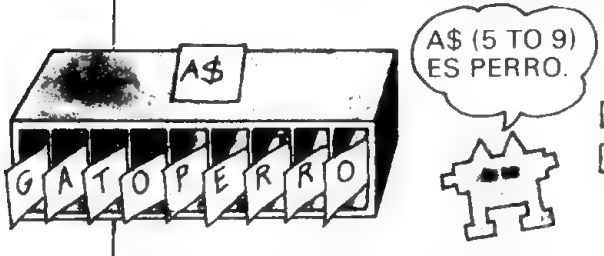
Esta manda a la computadora a imprimir la siguiente línea de datos.

La computadora necesita estos pasos para imprimir las líneas de datos y las palabras en el orden correcto. Esta sección del programa se repite cinco veces.

Cada vez la computadora imprime a el número de línea data de A y algunas palabras data del número de fila de B. Las palabras data que se eligen se deciden por el número aleatorio N.

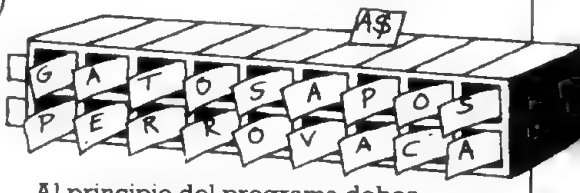
Computadoras y variables *SINCLAIR*

Este programa no funciona de esta manera en su *SINCLAIR* porque manejan las variables de cadena de forma distinta.



Para decir al *SINCLAIR* que elija un dato particular de una variable debes darle el número del primer y último carácter del elemento que desees. Este es el mismo sistema que el *SINCLAIR* emplea para *LEFT$*, *RIGHT$*, etc. (ver página 32).

Para matrices bidimensionales debes decir a la computadora el número de fila, así como el número de caracteres. Por ejemplo: A$(2,6 TO 9) es VACA.



Al principio del programa debes decir a la computadora cuántas filas tiene la matriz y cuántos caracteres hay por fila. Por ejemplo: *DIM* A$(2,9) quiere decir dos filas cada una de nueve caracteres. Todas las filas de la matriz deben tener el mismo número de caracteres.

Programa completo de cuentos

Ahora puedes escribir seguidas todas las partes del programa y escribir el programa del cuento completo. La primera parte del programa (pasos 10 a 190) dan los datos a la computadora y la segunda parte (pasos 200 a 310) imprimen el cuento; cada vez que se corre el programa obtiene una versión diferente del poema porque el número aleatorio N hace que el ordenador elija diferentes palabras.

```
 10 DIM A$(5)
 20 DIM B$(7,4)
 30 FOR I=1 TO 7
 40 FOR J=1 TO 4
 50 READ B$(I, J)
 60 NEXT J
 70 NEXT I
 80 DATA TASHKENT, TRENT, KENT, GHENT
 90 DATA ENVOLVIO, CUBRIO, PINTO, ATO
100 DATA CABEZA, MANO, PERRO, PIE
110 DATA EN UNA TIENDA, CON CEMENTO, CON ESENCIA, QUE ESTABA DOBLADO
120 DATA SE ESCAPO, LUCIO, SE VOLO, SE VOLVIO AZUL
130 DATA EN EL PARQUE, COMO UN CRISTAL, PARA UNA ALONDRA, CON UN LADRIDO
140 DATA DONDE FUE, SU INTENTO, PORQUE SE FUE, QUE QUERIA DECIR
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT (I)
180 DATA HABIA UN JOVEN DE, QUIEN, SU
190 DATA UNA NOCHE DESPUES DE OSCURECER, NUNCA ADIVINO
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=RND(1)*A+1
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

Los pasos 10 y 20 indican cuánto espacio se debe reseñar a las variables una fila de 5 para A$ y 7 filas de 4 para B$.

Estos son bucles anidados para poner los datos en B$.

Entre el paso 80 y el 140 están los datos que se almacenarán en B$.

Esto es un bucle para poner los datos en A$.

Los pasos 180 y 190 contienen las líneas de datos para almacenarse en A$.

Escribe la línea de datos almacenada en A$ en el compartimento A.

Escribe las palabras almacenadas en B$, fila B, columna N.

El programa se detiene en el paso 230 cuando A=6, así que nunca llega al paso 310, pero algunos computadores precisan el END.

Ejecuciones

```
HABIA UN JOVEN DE
KENT
QUIEN
ENVOLVIO
SU
CABEZA
EN UNA TIENDA
UNA NOCHES DESPUES DE OSCURECER
LUCIO
COMO UN CRISTAL
Y NUNCA ADIVINO
PORQUE SE FUE
```

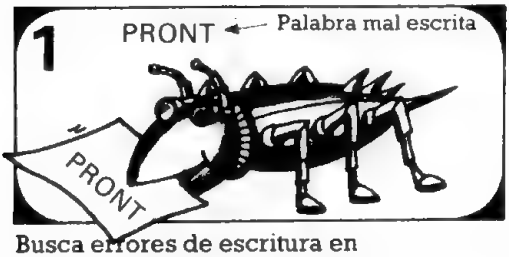
```
HABIA UNA JOVEN DE
GHENT
QUIEN
PINTO
SU
PIE
CON CEMENTO
UN NOCHE DESPUES DE OSCURECER
SE VOLVIO AZUL
COMO UN LADRIDO
Y NUNCA ADIVINO
SU INTENCION
```

Aquí hay dos versiones de las 16.389 disponibles. Si pruebas este programa y obtienes siempre el mismo cuento, mira en tu manual para ver como generar diferentes números aleatorios. Algunos ordenadores producen la misma secuencia de números aleatorios cada vez que se encienden.

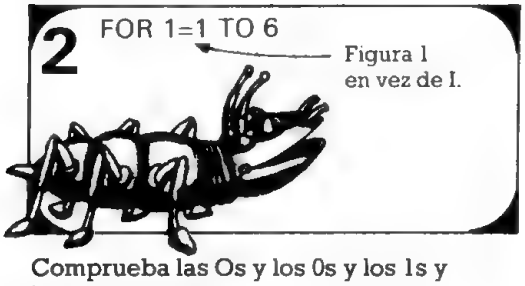
Avisos de programación

En estas dos páginas hay algunos avisos para ayudarte a escribir tus programas y una lista de los errores más comunes que pueden hacer y sus causas. Los errores más probables se enumeran primero; de manera que si tienes un programa que no funciona repasa esta lista hasta hallar el error.

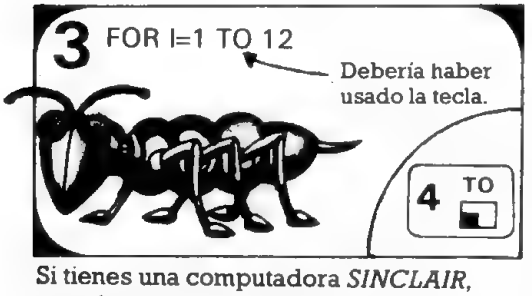
Encontrando errores



Busca errores de escritura en palabras *BASIC*. Si escribes mal una de ellas la computadora no la reconocerá.



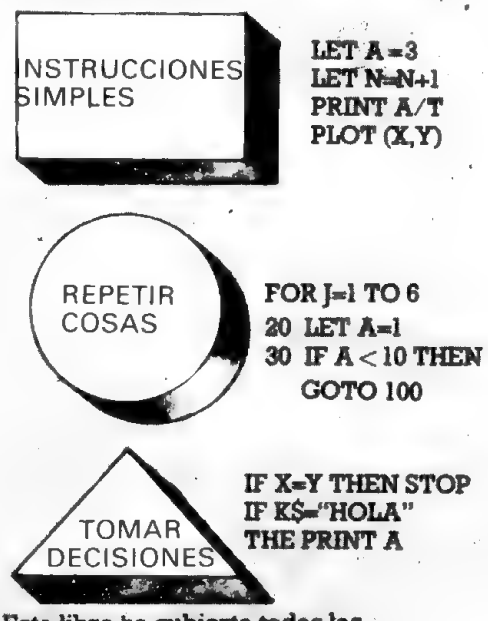
Comprueba las Os y los 0s y los 1s y las Is para asegurarte que has escrito lo correcto en los lugares correctos.



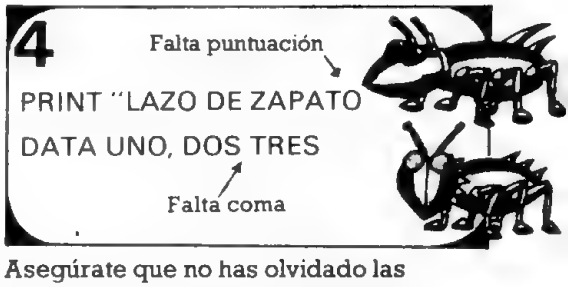
Si tienes una computadora *SINCLAIR*, asegúrate que no has escrito una palabra letra por letra en vez de apretar la tecla de esa palabra.

Escribiendo programas

Cuando estás escribiendo un programa es bueno recordar que la computadora puede llevar a cabo tres actividades principales: instrucciones simples, repetir cosas y tomar decisiones. Estos son los bloques de construcción de todos los programas.



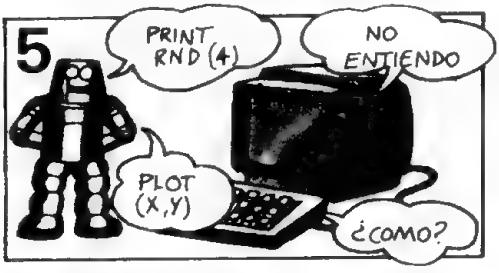
Este libro ha cubierto todas las instrucciones principales que precisas en *BASIC* para decirle a la computadora que lleve a cabo estas actividades. Cuando estás escribiendo un programa, calcula lo que la computadora debe hacer en cada paso.



Asegúrate que no has olvidado las comillas o las comas entre los datos. Comprueba los pasos complicados que tienen muchos símbolos con especial atención.

Generalmente hay varias formas diferentes de escribir un programa y algunas de ellas pueden ser más claras y cortas que otras. Cuando escribas un programa largo es una buena idea dividirlo en secciones con subrutinas para llevar a cabo las actividades. El cuerpo central del programa puede ser un simple conjunto de instrucciones, deciciones y repeticiones que controlen cuándo y cuántas veces la computadora lleva a cabo las subrutinas.

Partir los programas en secciones como estos facilita la búsqueda de errores. Cada sección puede probarse por sí sola generalmente sin ejecutar todo el programa. Acuérdate de asignar un nombre a cada sección con un paso *REM* para saber qué es.



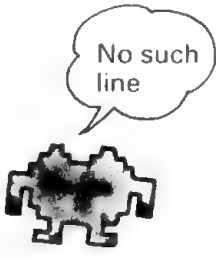
Asegúrate que usas los comandos RND, PLOT y CLS correctos para tu computadora. Comprueba también que has dado a la computadora un paso para gráfica si la necesita.

Mensajes de error

Todas las computadoras imprimen algún mensaje cuando hay un error en el programa, y los mensajes son explicados en el manual de la computadora. Aquí hay algunos de los más comunes mensajes que puedes obtener.

◄ Esto quiere decir que no hay suficientes datos para la lectura en los pasos *DATA*. Puede ser porque hayas olvidado una coma entre dos elementos y la computadora los haya tomado como uno solo.

► El paso dado con un número en un GOTO o *GOSUB* no existe. Puedes haber borrado el paso accidentalmente al escribir otro paso con el mismo número o puedes haber escrito mal el número.



◄ Puedes obtener esta nota en un *BBC* o en una computadora *SINCLAIR*. Quiere decir, generalmente, que no has escrito la variable en una instrucción *LET C=0* o *LET C=" "* antes de usarla.

► Esto quiere decir que el paso *NEXT* de un bucle falta. Puede ser porque has escrito el nombre de variable equivocado o has puesto un 1 en lugar de una *l*.

Ultima palabra

Algunos errores son muy difíciles de localizar, pero si la computadora no ejecuta el programa debe haber un error en algún sitio. Si realmente no puedes encontrarlo, intenta volver a escribir los pasos complicados otra vez, puedes obtenerlos bien la segunda vez, sin ni siquiera saber cuál era el error.

Respuestas a los ejercicios

Página 15
Programa: Nombre y mensaje

```
10 PRINT "COMO TE LLAMAS"
20 INPUT N$
30 PRINT "HOLA"
40 PRINT N$
50 PRINT "¿COMO ESTAS?"
```

Página 17
1. Programa: Operaciones

```
10 LET A=9
20 LET B=7
30 PRINT A★B
40 PRINT A/B
50 LET A=A+1
60 LET B=B+1
70 PRINT A★B,A/B
80 END
```

Coma para dejar espacio

2. Programa: Tablas

```
30 PRINT A; "POR"; B; "ES"; A★B
40 PRINT A; "DIVIDIDO POR"; B; "ES"; A/B
```

3. Alteraciones en nombre y mensaje

```
10 PRINT "COMO TE LLAMAS"
20 INPUT N$
30 PRINT "HOLA"; N$; "COMO ESTAS"
```

Página 18
Programa: Operaciones

```
10 PRINT "¿CUANTAS SON 7 POR 7?"
20 INPUT A
30 IF A=49 THEN PRINT "CORRECTO"
40 IF A< >49 THEN PRINT "NO"; 7★7
```

Necesitas punto y coma después de las comillas.

Página 19
Juego para adivinar la edad

Reemplazar paso 30 y añadir el 35:

```
30 IF G<14 THEN PRINT "MAYOR QUE"
35 IF G>14 THEN PRINT "MENOR QUE"
```

Página 23
Contador

```
 5 LET C=0
45 LET C=C+1
50 IF C< THEN GOTO 10
```

Dibuja tu incial

```
10 LET X=15
20 LET Y=30
30 PLOT (X,4)
40 LET 4=4-1
50 IF 4>5 THEN GOTO 30
60 LET X=X+1
70 PLOT (X,Y)
80 IF X<45 THEN GOTO 60
90 END
```

Página 24
Números Aleatorios

La fórmula para un número aleatorio entre 10 y 20 sería INT(RND(1)★11+9). En computadora, que sólo necesita un número entre paréntesis después de RND, sería RND(11)+9. Hay once posibles números entre 10 y 20, así que debes elegir números aleatorios entre 1 y 11 y luego sumar 9.

Página 25
Ataque espacial

Estos son los pasos que debes seguir para contar los aciertos:

```
15 LEFT S=0
75 IF X=A★B THEN LET S=S+1
95 PRINT "ACIERTO"; S; "DE 6 NAVES"
```

Página 27
1. Tabla del ocho

```
10 PRINT "TABLA DEL OCHO"
20 FOR J=1 TO 12
30 PRINT J; "X8="; J★8
40 NEXT J
```

Página 27
2. Tabla del número N

```
10 INPUT "TECLEE UN NUMERO"; N
20 PRINT "AQUI ESTA LA TABLA DEL"
   N
30 FOR I=1 TO 12
40 PRINT I; "VECES"; N; "ES"; I*N
50 NEXT I
60 INPUT "OTRO NUMERO (Y o N)"; M$
70 IF M$="Y" GOTO 10
```

Para el ZX81 necesitas los pasos *PRINT* e *INPUT* separados.

Página 32
Computer Book

LEFT$(A$,8) ES "COMPUTER"
RIGHT$(A$,10) ES "PUTER BOOK"
MID$(A$,5,8) ES "UTER BOO"

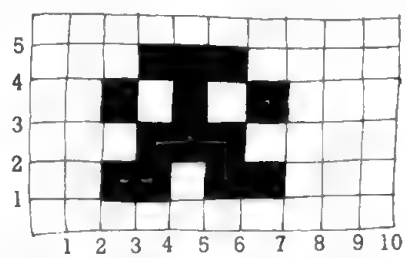
Página 34
Programa: Truco del número

```
10 PRINT "PIENSE UN NUMERO"
20 PRINT "DUPLIQUELO; SUMELE 4"
30 PRINT "DIVIDA POR 2; SUME 7"
40 PRINT "MULTIPLIQUE POR 8;
   RESTE 12"
50 PRINT "DIVIDA POR 4 Y QUITE 11"
60 PRINT "DIGAME EL RESULTADO"
70 INPUT N
80 PRINT "EL NUMERO QUE PENSO ES"
   (N-42)/2
```

Necesitas los paréntesis para hacer la suma en el orden correcto.

Programa repetido de los invasores espaciales

1



Dibuja el contorno simple de un invasor espacial en una hoja de papel cuadriculado.

2

4,5; 5,5; 6,5

3,4; 5,4; 7,4

4,3; 5,3; 6,3

3,2; 4,2; 6,2; 7,2

Ahora calcula las coordenadas de todos los cuadrados que constituyen el invasor.

3

```
5 CLS
50 INPUT "CUANTOS PUNTOS HORIZONTALES"; W
60 INPUT "CUANTOS VERTICALES"
65 CLS
70 FOR I=0 TO V STEP V/6
80 FOR J=0 TO W STEP W/6

130 NEXT J
140 NEXT I
150 END
```

Cambia el 6 a un número mayor para incrementar el número de veces que el invasor se repite en la pantalla. Si obtienes error es que has puesto un número demasiado grande.

Pon tus líneas de dibujo aquí, por ejemplo: 90 (PLOT(J+3,I+2). Para los dos cuadros de abajo a la izquierda.
92 PLOT(J+4,I+2). Necesitas una línea de programa por cada cuadro.

Copia el programa modelo repetido excluyendo los pasos 10 a 40 y de la 90 a la 140, como se muestra arriba. (Estos pasos producen el modelo aleatorio para el programa y, por tanto, no los necesitas.) Ahora pon tus propios pasos de PLOT en el programa entre los pasos 80 y 140 (puedes volver a numerar los pasos del programa). Para cada par de coordenadas debes añadir J a la primera figura e I a la segunda para repetir el invasor espacial.

Palabras *BASIC*

Aquí hay una lista de las palabras *BASIC* utilizadas en este libro con pequeñas explicaciones de lo que significan. Algunas de ellas, como CLS, no son standard en todas las computadoras, y éstas llevan una estrellita al lado. Si tienes un micro, deberás comprobar estas instrucciones en tu manual.

BREAK En algunas computadoras detiene el programa. Cuidado, sin embargo, en otras borra el programa entero de la memoria y deberás usar en su lugar otra palabra como *ESCAPE.*

CLS Borra la pantalla.

DATA Una lista de datos (palabras o números) para almacenar en las variables. Ver *READ.*

DIM Dice a la computadora cuánto espacio de memoria debe reservar para una variable. Ej.: *DIM* A$(5,4) quiere decir que la variable necesita cinco filas de cuatro columnas.

EDIT Permite alterar una línea del programa sin tener que escribirla entera de nuevo.

END Indica a la computadora el fin del programa. Algunas computadoras deben llevar siempre una sentencia *END;* otros, como el *BBC* y el *SINCLAIR,* no la necesitan.

FOR ... NEXT Introduce a la computadora en un bucle del programa repitiendo las instrucciones interiores al bucle un número fijo de veces.

GOSUB Obliga a la computadora a dejar la parte central del programa para ir a una subrutina y llevar a cabo una tarea especial.

GOTO Dice a la computadora que vaya a otro paso del programa (IR ... A).

IF THEN Compara datos (números, palabras o contenidos de variables) y hace cosas diferentes dependiendo del resultado.

INPUT Una forma de conseguir que la computadora te pida los datos mientras se está ejecutando el programa.

INT Convierte un número con punto decimal en un número entero ignorando la parte decimal. Ej.: *INT*(3,40)=3.

LEFT$* Dice a la computadora que haga algo con un número de caracteres de la izquierda de la tira. Ej.: *LEFT$(A$,4) toma cuatro caracteres desde la izquierda de A$.

LEN Da la longitud de una cadena, esto es el número de caracteres de una variable.

Palabras de la computadora

MATRIZ Un conjunto de variables que contiene varios elementos de datos.

BUG Error del programa.

CPU Unidad Central de Proceso, que controla las operaciones y hace el trabajo. Ej.: Comparar variables, sumar, etc.

CURSOR Una pequeña luz, cuadrada o de otra forma, que aparece en la pantalla e indica dónde debe imprimirse el siguiente carácter.

FLOW CHART Un cuadro indicando las operaciones principales necesarias en el programa. A menudo utilizado como ayuda para escribir programas.

GRAPHICS Formas de producir información visual sobre la pantalla.

KILOBYTES (K) Unidad de medida de la memoria de la computadora. 1 Kilobyte son 1024 bytes en la mayoría de los micros cada carácter ocupa 1 byte.

PROMPT Una interrogación u otro símbolo que aparece en la pantalla cuando la computadora pide más información después de una sentencia INPUT.

LET Pone la etiqueta de una variable en un espacio de memoria y pone dentro de ella alguna información. Ej.: *LET* N=4 o *LET* B$="GASTOS".

★ *LIST* Saca por pantalla un listado del programa.

★ *MID$*Dice a la computadora que haga algo con los caracteres del centro de una tira. Ej.: *MID$*(AS,4,3) tomas tres caracteres comenzando por la cuarta letra de A$.

NEW Borra el programa de la memoria de la computadora para el siguiente programa.

NEWLINE KEY Dice a la computadora que has terminado de escribir un paso de programa o un *INPUT*. Algunas computadoras tienen teclas *RETURN* o *ENTER*.

NEXT Ver *FOR*.

★ *PLOT* Dice a la computadora que ilumine un *pixel*. Ej.: *PLOT* (X,Y) enciende el *pixel* en la posición X horizontal e Y vertical.

PRINT Dice a la computadora que saque algo por pantalla.

★ *READ* La computadora lee la información de una línea *DATA* y la almacena en una variable. Ver *DATA*.

★ *READY* Algunas computadoras dicen esto cuando están listas para recibir más instrucciones.

REM La computadora ignora estos pasos que comienzan con *REM*, pero las saca en el listado. Son útiles para recordar lo que hacen las diferentes partes del programa.

RETURN Al final de una subrutina manda a la computadora a la instrucción siguiente a la que él dejó. Ver *GOSUB*.

★ *RIGHT$* Dice a la computadora que haga algo con los caracteres de la derecha de una tira. Ej.: *RIGHT$*(A$,4) quiere decir que tome los cuatro caracteres de la derecha de A$.

★ *RND* Elije un número aleatorio.

RUN Ordena que se ejecute el programa.

SQR Ordena que halle la raíz cuadrada de un número.

STEP Usado en bucles *FOR ... NEXT*. Ordena cuando debe repetirse el bucle.

STOP Se utiliza en un programa para ordenar que detenga la ejecución de éste.

THEN Ver *IF*.

★ *UNPLOT* Dice a la computadora que apage un pixel.

PIXEL Abreviatura de elementos de dibujo. Son los pequeños cuadraditos (diodos) que la computadora puede iluminar en la pantalla para hacer dibujos.
PROGRAM Una lista numerada de instrumentos que dice a la computadora cómo llevar a cabo una tarea en particular.
RAM: Random Access Memory (Memoria de acceso directo). Esta memoria es donde se almacenan el programa y los datos. Esta información se borra al apagar la computadora.
ROM: Read Only Memory (Memoria de lectura sólo). Memoria permanente donde se almacena la información necesaria para que la computadora trabaje.
STRING Una serie de caracteres que se almacenan en una variable.
Ej.: "SALCHICHAS" o "ABC 123".
SUBRUTINA Una sección del programa que lleva a cabo una tarea específica que se suele repetir varias veces al correr el programa.
SYNTAX ERROR Errar en las palabras *BASIC* del programa.
VARIABLE Un espacio de memoria etiquetado que contiene información.

Yendo más allá

La mejor forma de aprender a escribir programas es probar programas en una computadora. Si no tiene uno, puede haber algún lugar donde puedas ir a usarlo. Pide en tu escuela, o en la biblioteca o en algún grupo de usuarios, si puedes usar durante algunas horas la computadora haciendo prácticas.

Puedes aprender mucho sobre como escribir programas a través de la lectura de programas escritos por otras personas. Aquí hay una lista de libros sobre computadoras y programación que puede serte útil.

PROGRAMACION DE COMPUTADORAS
MICRO COMPUTADORAS
CALCULADORAS DE BOLSILLO

MANUAL DE GRABACION
COMPUTADORAS
TV Y VIDEO

Todos ellos publicados por Ediciones Plesa (Madrid).

Indice

Los números en color más **oscuro** indican dónde se explican los términos por primera vez.

C/ Sesta, núm. 1. Pinto - Madrid - ESPAÑA.
Impreso en España - Printed in Spain. MELSA, PINTO - MADRID.
Depósito legal: M-34686-1983
I.S.B.N. 84-7374-109-9

Colección Electrónica

Los primeros libros de esta sorprendente colección introducen al lector en el maravilloso mundo de las calculadoras, computadoras y magnetofones a cassette. Las claras y coloridas ilustraciones llevan a lector al conocimiento tanto de los conceptos básicos y técnicas como de los más avanzados proyectos e ideas. Los libros contienen ejercicios, juegos y entretenidas actividades de fácil comprensión.

Calculadoras de Bolsillo
Contiene muchos ejercicios y juegos, explica cómo manejar las calculadoras sencillas y las científicas. Hay también problemas para ejercitar y practicar.

Manual de Grabación
Ofrece ideas para hacer grabaciones divertidas, incluyendo los sonidos de la naturaleza y los efectos especiales. Explica los trucos y técnicas para obtener grabaciones casi profesionales.

Juegos de computadora
Sencillo y entretenido conocimiento de cómo juegan las computadoras a los invasores espaciales, al ajedrez y a otros muchos juegos. Enseña trucos para vencer a la computadora.

Micro computadoras
Es una guía ilustrada sobre micro computadoras, de cómo trabajan y lo que pueden hacer. Enseña muchas cosas para hacer con una micro.

Programación de Computadoras
Es una guía sobre los programas en BASIC para principiantes. Enseña, paso a paso, muchos programas para hacer cualquier micro computadora.